UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA MECÂNICA

TÉCNICA DE NAVEGAÇÃO DE UM ROBÔ MÓVEL BASE-ADO EM UM SISTEMA DE VISÃO PARA INTEGRÁ-LO A UMA CÉLULA FLEXÍVEL DE MANUFATURA

CARLOS ENRIQUE VILLANUEVA CANO

ORIENTADOR: SADEK CRISOSTOMO ABSI ALFARO
CO-ORIENTADOR: ALBERTO JOSÉ ÁLVARES

DISSERTAÇÃO DE MESTRADO EM SISTEMAS MECATRÔNICOS

PUBLICAÇÃO: DM - 06/2006
BRASÍLIA/DF: MAIO – 2006

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA MECÂNICA

TÉCNICA DE NAVEGAÇÃO DE UM ROBÔ MÓVEL BASEADO EM UM SISTEMA DE VISÃO PARA INTEGRÁ-LO A UMA CÉLULA FLEXÍVEL DE MANUFATURA (FMC)

CARLOS ENRIQUE VILLANUEVA CANO

DISSERTAÇÃO SUBMETIDA AO DEPARTAMENTO DE ENGE-NHARIA MECÂNICA DA FACULDADE DE TECNOLOGIA DA U-NIVERSIDADE DE BRASÍLIA COMO PARTE DOS REQUISÍTOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM SISTEMAS MECATRÔNICOS.

APROVADA POR:

____

Prof. Sadek Crisóstomo Absi Alfaro, PhD. (ENM-UnB)
(Orientador)

____

Prof. José Maurício S. T. da Motta, PhD. (ENM-UnB)
(Examinador Interno)

____

Prof. Álisson Rocha Machado, PhD. (UFU)
(Examinador Externo)

BRASÍLIA/DF, 12 DE MAIO DE 2006

FICHA CATALOGRÁFICA

CANO, CARLOS ENRIQUE VILLANUEVA
Técnica de Navegação de um Robô Móvel Baseado em um Sistema de Visão para Integrá-lo em uma Célula Flexível de Manufatura (FMC) [Distrito Federal] 2006.
xx, 154p., 210 x 297 mm (ENM/FT/UnB, Mestre, Engenharia Mecânica, 2006).
Dissertação de Mestrado – Universidade de Brasília. Faculdade de Tecnologia.
Departamento de Engenharia Mecânica.

1. Célula Flexível de Manufatura   2. AGV
3. Navegação Robótica   4. Visão Computacional
I. ENM/FT/UnB   II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

CANO, C. E. V. (2006). Técnica de Navegação de um Robô Móvel Baseado em um Sistema de Visão para Integrá-lo em uma Célula Flexível de Manufatura (FMC). Dissertação de Mestrado, Publicação DM-06/2006, Departamento de Engenharia Mecânica, Universidade de Brasília, Brasília, DF.,154p

CESSÃO DE DIREITOS

AUTOR: Carlos Enrique Villanueva Cano.
TÍTULO: Técnica de Navegação de um Robô Móvel Baseado em um Sistema de Visão para Integrá-lo em uma Célula Flexível de Manufatura (FMC).
GRAU: Mestre   ANO: 2006

Carlos Enrique Villanueva Cano
SHIN QL. 03 CONJ. 05 CASA 19, Lago Norte
71505-255 BRASILIA - DF - Brasil

Dedicatória

Dedico este trabajo a toda mi familia, mi mamá Aurora, mi hermana Mónica mi sobrinita Ani y muy especialmente a mi padre: El Señor ***Samuel Bernardino Villanueva Guío*** quien a lo largo de toda mi vida como estudiante siempre ha estado presente alentándome y sobretodo acompañándome en los momentos decisivos; y así como él estuvo presente en este paso importante de mi vida, ojalá continúe estando siempre en los pasos futuros.

Agradecimentos

Va el mejor de mis agradecimientos al Profesor ***Sadek C. Absi A***. por su apoyo, comprensión y sobre todo por el trato de amigo a lo largo del periodo de desenvolvimiento de este trabajo.

Me faltarían palabras para agradecer a todas las personas que conocí a lo largo de mi estadía aquí en Brasil, personas amigas: Laercio Jardim, Andre Braga, Daniel Muñoz, Magno Correia, Flavio de Barros Vidal, Evandro Texeira, Leonardo "Junior", etc. personas de trabajo: El Sr. Marrocos e los trabajadores de la SG9, especialmente al Sr. Claudio Pereda; profesores amigos: Carlos Llanos Quinteros, João Carlos (UFSC), Martin (UFSC), Guilherme Caribe Carvalho, José Maurício Motta e Profesor Álberto Alvares.

También quiero agradecer de manera muy especial a las siguientes personas amigas: Marveline, Sra. Grimalda, Elisa, Jesús, Sadek "Junior" e José Vargas entre otros, con quienes viví momentos agradables aquí en Brasil.

Muchas Gracias a todos Uds., sin duda me llevo el mejor de los recuerdos de todos al Perú. Como dirían los Braslileños: "*Valeu Galera*".

Hasta pronto.

RESUMO

TÉCNICA DE NAVEGAÇÃO DE UM ROBÔ MÓVEL BASEADO EM UM SISTEMA DE VISÃO PARA INTEGRÁ-LO EM UMA CÉLULA FLEXÍVEL DE MANUFATURA (FMC)

Autor: Carlos Enrique Villanueva Cano

Orientador: Sadek Crisóstomo Absi Alfaro

Programa de Pós-graduação em Sistemas Mecatrônicos

Brasília, maio de 2006

Atualmente o controle dos veículos guiados automaticamente (AGV), é baseado no estudo das diferentes técnicas de navegação existentes. Estas técnicas, utilizam diferentes tipos de sensores (infravermelho, ultra-som, táctil, visão, etc) como parte do equipamento do robô móvel para realizar suas tarefas de movimentação e controle desde uma posição inicial até uma posição final definida.

A contribuição principal deste trabalho consiste em apresentar uma técnica de navegação baseada em um sistema de visão, para integrar um robô móvel à célula flexível de manufatura (FMC), a qual está sendo implementada no laboratório GRACO – UNB, trabalhando como um veículo guiado automaticamente (AGV), assim como também apresentar as vantagens e desvantagens do tipo de navegação implementada.

O sistema de navegação proposto é composto por cinco módulos principais: O primeiro módulo é da captura de imagens no formato RGB através da placa de captura Matrox Meteor; O segundo módulo de pré-processamento, que realiza o tratamento das imagens capturadas através da aplicação de algoritmos de visão computacional. O terceiro módulo de extração de pontos das linhas seguidas, que extrai da imagem binarizada os pontos mais importantes da cena como é o caso da extração de dois pontos da linha para calcular a sua inclinação para depois ser comparada com os erros mínimos permitidos. O quarto módulo é o de reconhecimento dos sinais através da utilização de algoritmos baseados em redes neurais; e finalmente o quinto módulo é o de controle, que com base à informação gerada nos módulos anteriores, trabalha nos motores do robô para gerar ou corrigir os movimentos ao longo da linha rastreada ou executar uma ação de movimento dependendo do sinal reconhecido. O sistema de navegação utilizado, pelo robô móvel, é modelado utilizando técnicas de objetos orientados, através da linguaje UML.

ABSTRACT

MOBILE ROBOT NAVIGATION TECHNIQUE BASED IN VISION SYSTEM FOR INTEGRATE IT TO A FLEXIVEL MANUFACTURING CELL (FMC)

Author: Carlos Enrique Villanueva Cano

Supervisor: Sadek Crisóstomo Absi Alfaro

Programa de Pós-graduação em Sistemas Mecatrônicos

Brasília, May of 2006

Nowadays, the control of the automatic guided vehicles (AGV), is based on the study of different navigation existing techniques. Those utilize different kinds of sensors (infrared, ultrasound, tactile, vision, etc) as part of the equipment included in the mobile robot carrying out their tasks of movement and control since an initial position to a defined final position.

The main contribution of this work is developing a navigation technique based on a vision system, by integrating a mobile robot to a flexible manufacturing cell (FMC), that is being implemented on the GRACO – UnB laboratory. On the mobile robot, working as an automatic guided vehicle (AGV), are shown, as well, the advantages and disadvantages of this kind of navigation.

The system of proposed navigation is building up for five main modules: The first module is images capture module in RGB format through the Matrox Meteor board. The second module of pre-processing, carries out the processing of the captured images through the application of computational vision algorithms. The third is the line points extraction module, to extract from the image the most important points of the scene as is in the case of the slope line tracking calculation and comparing it with the minimum permitted errors in the tracking line. The fourth module of signs recognition through the utilization of algorithms based in neural nets ; and finally the fifth module of control, that base on the information generated in the previous modules, works in the motor control of the robot, generating the correct movements during the tracking line or executing an action of movement depending on the recognized sign. The navigation system utilized by the mobile robot will be modelled, utilizing techniques of objects oriented, through the UML language.

RESUMEN

TÉCNICA DE NAVEGACIÓN DE UN ROBOT MÓVIL BASADO EN UM SISTEMA DE VISIÓN PARA INTEGRARLO EN UNA CÉLULA FLEXIBLE DE MANUFACTURA (FMC)

Autor: Carlos Enrique Villanueva Cano

Orientador: Sadek Crisóstomo Absi Alfaro

Programa de Pós-graduação em Sistemas Mecatrônicos

Brasília, mayo del 2006

Actualmente el control de los vehículos guiados automáticamente (AGV), esta basado en el estudio de las diferentes técnicas de navegación existentes. Estas, utilizan diferentes tipos de sensores (infrarrojos, ultrasonido, táctiles, visión, etc.) como parte del equipo del robo móvil para realizar sus tareas de navegación y control, desde una posición inicial hasta una posición final definidas.

La principal contribución de este trabajo consiste en presentar una técnica de navegación basada en un sistema de visión para integrar un robot móvil a una célula flexible de manufactura, a cual está siendo implementada en el laboratorio de automatización y control GRACO-UnB, trabajando como un vehículo guiado automáticamente, así como también presentar las ventajas y desventajas del tipo de navegación que esta siendo implementada.

El sistema de navegación propuesto está compuesto por cinco módulos principales: un primer módulo de captura de imágenes en formato RGB a través da placa Matrox Meteor de el robot móvil. Un segundo módulo de pre-procesamiento de las imágenes capturadas a través de la aplicación de algoritmos de visión computacional. Un tercer módulo de extracción de puntos de las líneas seguidas por el robot para calcular su inclinación a cual será comparada posteriormente con los errores mínimos permitidos. O cuarto módulo de reconocimiento de señales a través del uso de algoritmos basados en redes neurales; y finalmente el quinto módulo de control, el cual en base a la información generada por los módulos anteriores, trabaja en el control de los motores del robot los cuales generan el correcto desplazamiento durante el rastreo de las líneas o ejecutan alguna otra acción de movimiento dependiendo de la señal reconocida. El sistema de navegación que está siendo implementado en el robot móvil será modelado utilizando técnicas de objetos orientados a través del uso del lenguaje UML.

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE SÍMBOLOS, NOMENCLATURA E ABREVIAÇÕES

$p(r_k)$	- Histograma de uma imagem
r_k	- Transformação inversa – Equalização de histograma
$P_A(x,y)$	- Valor atualizado do pixel
$P(x,y)$	- Valor original do pixel em (x,y)
$M(i,j)$	- Valor da máscara na posição (i,j)
i, j	- Variáveis do somatório
∇f	- Gradiente de f nas coordenadas (x,y)
$\Delta^2 I(x,y)$	- Operador Laplaciano
W	- Real world space
I	- Espaço da imagem 2D
S	- Plano xy do espaço W que será definida como a superfície
$\bar{\gamma}$	- É o ângulo formado pela projeção do eixo ótico $\hat{o}$ no plano $z = 0$ e o eixo x
$\hat{\eta}$	- Vetor unitário
$\bar{\theta}$	- Ângulo formado pelo eixo ótico $\hat{o}$ e o vetor unitário $\hat{\eta}$ como se mostra na
$\hat{o}$	- Eixo ótico
2α	- Abertura angular da câmera
n x n	- Resolução da câmera é
h_c	- Altura da câmera
β	- Ângulo da câmera
n_v	- Número de pixéis verticais
n_h	- Número de pixéis horizontais
$hCCD$	- Tamanho horizontal do pixel da câmera
$vCCD$	- Tamanho vertical do pixel da câmera
f	- Distância focal da câmera dado pelo fabricante

α_h	- Abertura angular vertical da câmera
α_v	- Abertura horizontal
$\mathcal{N}$	- Definição de uma rede neural
ALVINN	- Autonomous Land Vehicle in a Neural Net
AHVS	- Automated Highway Vehicle System
ALV	- Autonomous Land Vehicle
AGV	- Automatic Guided Vehicle
AS / RS	- Automatic Storage / Retrieval System
CAD	- Projeto Auxiliado por Computador
CAPP	- Planejamento de Processo Auxiliado por Computador
CAM	- Manufatura Auxiliada por Computador
Cfree	- Espaço livre
CIM	- Manufatura Integrada por Computador
CLP	- Controlador Lógico Programável
CNC	- Comando Numérico Computarizado
DARPA	- Defence Advanced Research Projects Agency
DSP	- Digital Signal Processing
FIPMA	- Fast inverse perspective mapping algorithm
FMC	- Flexible Manufacturing Cell
FMS	- Flexible Manufacturing System
FOCAS 1	- Fanuc Open CNC API Specifications
FOV	- Field of View
FTP	- File Transfer Protocol
frame graber	- Captura de imagem
GOLD	- Generic Obstacle and Lane Detection
GPS	- O Sistema de Posicionamento Global, vulgarmente conhecido por - GPS (do acrónimo do inglês Global Positioning System)
GRACO	- Grupo de Automação e Controle da Universidade de Brasília
GUI	- Interface Gráfica do Usuário
HG	- Geração de hipóteses
HTTP	- Hypertext Transfer Protocol
HV	- Verificação de hipóteses
HSV	- Hue, Saturation, Value)

IDEF	- Integration Definition for Function Modeling
IPM	- Inverse Perspective Mapping
ITS	- Intelligent transportation systems
I/O	- Input / Output
IVVI	- Intelligent Vehicle base on Visual Information
MLP	- Multilayer perceptrons
MgU	- Management Unit
MARF	- Maryland Road Following
MODT	- Multiple object detector
NC	- Comando Numérico
NTSC	- National Television Standards Committee
PVS	- Personal Vehicle System
PROMETHEUS	- Program for a European Traffic with Highest Efficiency and Un-precedented Safety
RP	- Redes de Petri
RGV	- Rail Guided Vehicle
RGB	- Red, Green and Blue
RDT	- Road Detection and Tracking
RALPH	- Rapidly Adapting Lateral Position Handler
SODT	- Single object detector and tracker
TF(R)	- A transformada de Fourier da convolução $R(x, y)$
TCP / IP	- TCP - Transmission Control Protocol (Protocolo de Controle de Transmissão) - e o IP - Internet Protocol.
TG	- Tecnologia de Grupo
UML	- Unified Modeling Language
WWW	- World Wide Web – Rede Mundial de Computadores
XRDEV	- Arquitetura multi-processada do Robo Movel Nomad XR 4000

1 – INTRODUÇÃO

1.1 - ESTADO TECNOLÓGICO

Com o desenvolvimento de tecnologias de navegação para veículos autônomos e o aumento da capacidade de processamento dos computadores, apareceram os primeiros robôs moveis industriais. Embora o seu âmbito de aplicação seja muito mais amplio, segundo Rocha (2001), um robô móvel procura conjugar num só dispositivo de automação a mobilidade de um veículo autônomo e a capacidade de manuseamento e manipulação dos robôs. Neste domínio, o *AGV* (*Automatic Guided Vehicle*) é um dispositivo com maior aplicação em empresas industriais ou de distribuição.

Nos últimos anos houve muito interesse no desenvolvimento de tecnologias aplicadas em veículos guiados automaticamente. Desde a automatização de tarefas que envolvem traslado e carga de materiais ou simples tarefas de inspeção, onde implique o movimento desde um ponto inicial a um ponto final do veículo, tem havido uma grande melhoria na redução de riscos, tempos de traslado e consumo de energia.

A movimentação dos veículos guiados automaticamente (*AGV*), é baseada no estudo das diferentes técnicas de navegação existentes, as quais utilizam diferentes tipos de sensores (infravermelho, ultra-som, táctil, visão, etc) como parte do equipamento do robô móvel para realizar tarefas de movimentação do robô desde uma posição inicial até uma posição final.

Os veículos guiados automaticamente são usados para transporte interno e externo de materiais. Tradicionalmente os *AGV's* foram os mais usados em sistemas de manufatura; atualmente os *AGV's* são utilizados para tarefas de transporte repetitivas e em outras áreas tais como armazenagem, sistemas de transporte externo (subsolo), etc.

O uso dos *AGV's* tem crescido enormemente desde sua introdução. O número de áreas de aplicação e variação nos tipos tem aumentado significativamente. Armazéns e centros com muitas intersecções são exemplos de áreas distribuídas.

Existem três tecnologias as quais são comumente usadas em sistemas comerciais dos *AGV's*: (1) cabo guiado, (2) rastreamento de caminhos desenhados no chão "*paint strips*" e (3) veículos auto-guiados. A tecnologia mais recente (laser, GPS, fusão sensorial, visão, etc.), desenvolvida para navegação robótica e especificamente para o caso dos *AGV's*, é utilizada nos veículos auto-guiados.

Assim o desenvolvimento de robôs móveis é uma tarefa fortemente interdisciplinar, envolvendo áreas tecnológicas tão diversas como: sensores e atuadores, eletrônica de potência, energia, projeto mecânico, cinemática, dinâmica, teoria de controle, escalonamento em tempo real, investigação operacional, sistemas de informação, telecomunicações, etc.

A nível internacional, o campo de aplicação dos robôs móveis não é só na indústria, este é significativamente mais amplo abarcando também as áreas de logística (distribuição e armazenagem), exploração subaquática e oceanográfica, exploração planetária bem como aplicações militares.

Atualmente na industria e especificamente em Portugal, segundo Rocha (2001), os projetos industriais de robótica móvel já realizados tiveram aplicação principalmente na industria (fábricas, células e sistemas flexíveis de manufatura), na logística de cadeias de distribuição e armazenagem e nos serviços. Neste tipo de aplicações destacam-se dois tipos de dispositivos: o *AGV (Automatic Guided Vehicle) e o AS / RS (Automatic Storage / Retrieval System).*

1.2 – OBJETIVO DO TRABALHO

Neste trabalho propõe-se a implementação de um dos métodos de navegação, para resolver o problema de planejamento da trajetória de um robô móvel e assim poder integrá-lo a uma célula flexível de manufatura (*FMC*) que está sendo implementada no laboratório (GRACO – UnB). O robô móvel estará trabalhando como um veículo guiado automaticamente (*AGV*), cujo sistema de navegação apresentará vantagens e desvantagens, as quais também serão analisadas neste trabalho.

O sistema proposto de navegação será modelado utilizando técnicas de orientação de objetos através da linguaje *UML* (*Unified Modeling Language*). A modelagem utilizando objetos orientados fornece um novo ponto de vista para focar um sistema flexível de manufatura usando modelos organizados, como são os módulos flexíveis de manufatura, os quais compõem uma *FMC*.

Este método de navegação é baseado no sistema de visão do robô móvel, e é uma alternativa econômica que pode ser implementada em ambientes estruturados (ambientes limpos,

sem movimentos ou deslocamentos constantes de pessoas ou outros equipamentos nos caminhos pintados no chão de fábrica).
A célula flexível de manufatura que está sendo implementada cumpre com os requisitos necessários de um ambiente estruturado, garantindo um bom funcionamento do sistema de navegação. Por outro lado, em um sistema flexível de manufatura com muitas estações e fluxo de produção alto e variável, o sistema de navegação que está sendo implementado poderia ter dificuldades, quando o objetivo seja criar novas rotas de aceso nos diferentes lugares do chão de fábrica.

Outro objetivo é a implementação física da *FMC* relacionado aos sistemas de comunicação entre os diferentes componentes da célula, assim como os de transporte, manuseamento e armazenamento de materiais.
Verifica-se como principal beneficio desta implementação o baixo custo tanto na parte física como na parte de programação do robô.

1.3 – RESUMO DOS CAPÍTULOS

No capítulo 2 é apresentada uma revisão bibliográfica detalhada das tecnologias iniciais e recentes que estão envolvidas neste trabalho.

No capitulo 3 são apresentados os conceitos fundamentais que estão envolvidos numa manufatura flexível. Estes conceitos são importantes para entender o ambiente onde um *AGV* realiza suas funções como transportador de material, e como o trabalho realizado por ele é fundamental para dar validez ao termo "flexível" o qual estabelece a diferença em uma manufatura automatizada.

No capítulo 4 são apresentados com detalhes os conceitos que estão envolvidos nos manipuladores e transportadores de material, como um dos componentes principais da *FMC*.

No capítulo 5 é apresentado o estado da arte da robótica móvel aplicada nos *AGV's*. Este ponto é fundamental para conhecer os diferentes tipos de sistemas de navegação usados atualmente em robótica móvel e especificamente nos *AGV's* industriais.

No capítulo 6 são apresentados, de forma geral, os diferentes tipos de navegação robóticas utilizados atualmente, não só para sistemas de navegação de *AGV's* mas também para qualquer aplicação de robótica móvel.
No capítulo 7 são apresentados os diferentes métodos de processamento de imagens baseados em visão computacional necessários para desenvolver os algoritmos do sistema de navegação do robô móvel que trabalhará como *AGV* na *FMC*.

No capítulo 8 são apresentadas as características básicas do hardware e software do robô móvel que atuará como *AGV* na *FMC*.

No capítulo 9 é apresentada a parte experimental deste trabalho. Aqui estão envolvidas as secões referentes a modelagem e implementação da *FMC*, como ambiente de atuação do *AGV*, a modelagem e implementação do sistema de navegação usado pelo robô móvel, para trabalhar como *AGV*, usando técnicas de orientação de objetos e algoritmos de visão computacional.

Finalmente, no capítulo 10 são apresentados os resultados experimentais obtidos, o capítulo 11 apresenta a discussão dos mesmos e no capítulo 12 são apresentadas algumas sugestões para trabalhos futuros.

2 – REVISÃO BIBLIOGRÁFICA

2.1 – HISTORIA

O campo de estudo onde gira o trabalho desenvolvido: integração de um robô móvel em uma *FMC*, se refere aos sistemas flexíveis de manufatura. Neste trabalho, especificamente, é estudado o desenvolvimento de um sistema de navegação e controle, referente ao sistema de transporte e manuseio de material (*material handling*) como um dos componentes principais de uma *FMC*. A movimentação dos veículos guiados automaticamente (*AGV*) é baseada no estudo das diferentes técnicas de navegação existentes, as quais utilizam diferentes tipos de sensores (infravermelho, ultra-som, táctil, visão, etc) como parte do equipamento do robô móvel para realizar tarefas de movimentação do robô desde uma posição inicial até uma posição final.

Um dos primeiros protótipos, baseados em visão para controlar a direção de um veículo foi apresentado in 1979 por um laboratório do governo Japonês. Este veículo tinha uma câmera de TV e um hardware exclusivo para o processamento de imagens. O sistema de visão tinha como tarefa procurar marcas brancas no caminho percorrido e assim conseguir o controle da direção e movimento do veículo (Masaki, 1992).

O Controle de um veículo baseado no sistema sensorial para ambientes diferentes a uma estrada "*off-road*" foi estudado como parte da *DARPA* (*Defence Advanced Research Projects Agency*), neste projeto esteve incluída também o projeto *ALV* (*Autonomous Land Vehicle*) que começou em 1984. Depois destes projetos diferentes trabalhos foram feitos com enfoque na percepção e planejamento de uma navegação autônoma.

Entre outras instituições, as Universidades de Massachusetts e Honeywell desenvolveram um software de rastreamento de movimento (*SRI*) e demonstrou um rastreamento usando dados em 3-D. Martin Marietta em 1984 construiu e operou o veiculo *ALV* e desenvolveu seu próprio software para seguimento de caminhos (*road-following software*). Hughes e a Universidade de Maryland contribuíram no desenvolvimento de sistemas de navegação *off-road* e *on-road* respectivamente no desenvolvimento do *ALV*. Já o aporte realizado pela Universidade de Carnegie Mellon foi o desenvolvimento de sistemas de visão a cor.

Depois dos trabalhos desenvolvidos pela *DARPA* houve outros projetos de veículos autônomos desenvolvidos nos cinco anos seguintes. Nos Estados Unidos, Texas *A&M* trabalhou com sistemas de rastreamento visual e desvio de obstáculos. A *General Motors* trabalho com sistemas de seguimentos de linhas a baixas velocidades com uma iluminação constante do ambiente de trabalho. Já a Universidade da Califórnia-Berkeley desenvolveu vários protótipos para o controle de direção baseado num sistema sensorial do veículo.

Na Europa, os veículos autônomos, baseados num sistema de visão e utilizados em rodovias, foram desenvolvidos com o apoio de *PROMETHEUS* (*Program for a European Traffic with Highest Efficiency and Umprecedented Safety*). O grupo de pesquisa da Universidade de München desenvolveu um sistema para guiar os veículos até 100 km/h em rodovias e a 50 km/h em caminhos onde as velocidades permitidas sejam mais baixas.

No Japão umas das primeiras companhias a desenvolver veículos autônomos, usando sistemas de visão, foram a *Nissan* junto com a companhia *Fujitsu*, as quais construíram um protótipo de veículos com sistemas para rastreamento de caminhos (Masaki, 1992).

2.2 - ALGORITMOS DE VISÃO

Para controlar os movimentos do veículo é suficiente interpretar só algumas partes da imagem que contém características importantes dela, não sendo necessário analisar todas as outras partes da imagem. Geralmente, só pequenas áreas da imagem contem características relevantes, onde deve estar concentrada toda a potência do processamento, que deve ser eficiente. A Figura 2.1 ilustra este conceito aplicado ao controle da direção de um veículo em uma rodovia. Comumente um sistema de controle de movimento baseado em visão utiliza menos do 10% do total da área da imagem, otimizando assim os recursos de processamento.

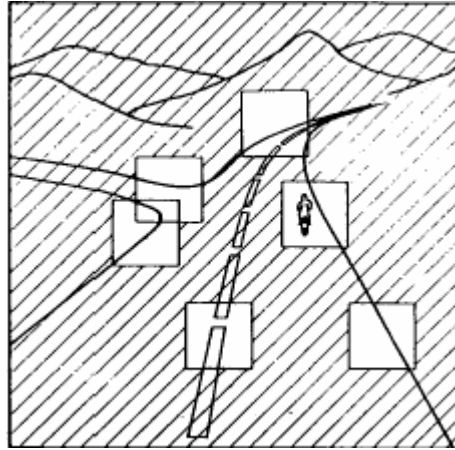



Figura 2.1 - Foco de atenção (Masaki, 1992))

A Universidade de München desenvolveu um sistema de visão para predizer a posição dos contornos dos caminhos (ou linhas). Outro aporte importante no desenvolvimento de algoritmos de visão foi feito pela Universidade de Maryland, com o projeto *MARF (Maryland Road Following)* que consistiu no desenvolvimento de um sistema através de visão para o seguimento de caminhos com ajuda basicamente de dois módulos, o módulo de planejamento, que decide que objetos devem ser procurados na imagem e como localizá-los, e o módulo de verificação responsável pelo controle do sensor e pelo processamento dos dados obtidos e verificação das predições feitas pelo módulo de planejamento.

A universidade de Osaka, baseada no conceito *optokinetic nystagamus*, referente a olho humano, desenvolveu um sistema de navegação para veículos autônomos para predizer as posições das bordas baseando-se no modelo de fluxo da imagem. As bordas preditas são comparadas com as bordas observadas. Estas comparações produzem resultados que são usados para atualizar o modelo de fluxo da imagem no sistema (Masaki, 1992)).

2.3 - REDES NEURAIS PARA PROCESSAMENTO DE IMAGENS

Um dos primeiros trabalhos realizados neste campo foi o *ALVINN* (*Autonomous Land Vehicle in a Neural Net*) desenvolvido pela Universidade Carnegie Mellon, o qual é um exemplo de um sistema de visão baseado em redes neurais para o controle de veículos autônomos. Este sistema controla os movimentos do veículo baseado na informação obtida por uma câmera de TV. Os pesos da rede neural são treinados manualmente e trabalha com

resoluções reduzidas de imagens de 30 por 32 ou de 45 por 48 pixéis. A desvantagem deste sistema é que foi treinado para rodar numa estrada particular, porem fácil de ser feito.

2.4 – PROCESSADORES DE VISÃO DE ALTA VELOCIDADE

Uma característica significante do processamento através de visão é a grande quantidade de dados envolvidos no processamento. A velocidade de processamento do algoritmo para a filtragem de uma só imagem é de aproximadamente 3×10^8 ciclos por segundo. Esta velocidade de processamento é até dez vezes maior que a velocidade de um microprocessador comum. Para solucionar estes problemas, basicamente dois aspectos foram pesquisados:

- Desenvolvimento de algoritmos compactos
- Desenvolvimento de processadores de alta velocidade

Pesquisas desenvolvidas até o ano 1992 mostraram que existem pelo geral dois tipos de arquiteturas para processadores de alta velocidade: (1) processadores altamente programáveis e (2) processadores de funções orientadas (Masaki 1992).

2.5 – TRABALHOS DESENVOLVIDOS NOS ÚLTIMOS ANOS

Nos últimos anos houve muito interes no desenvolvimento de tecnologias aplicadas em veículos guiados automaticamente. Desde a automatização de tarefas que envolvem traslado e carga de materiais ou simples tarefas de inspeção, onde implique o movimento desde um ponto inicial a um ponto final do veículo, tem havido uma grande melhoria na redução de riscos, tempos de traslado e consumo de energia.

Nesta parte do trabalho serão apresentados os resumos das mais recentes pesquisas desenvolvidas na automação de veículos para estradas (*Autonomous Road Following*), fazendo uma revisão das soluções experimentais e protótipos desenvolvidos mundialmente. As técnicas desenvolvidas nestes trabalhos, realizados até o ano 2000, são baseadas num sistema de visão e são as que serão usadas na proposta de integrar o robô móvel na *FMC*.

Os resultados mais interessantes e tendências neste campo, assim como a evolução dos veículos inteligentes nas próximas décadas são apresentados como resumos dos trabalhos realizados por (Bertozzi, et al., 2000).

Vários grupos de pesquisa em aplicações de sistemas de transporte inteligente ("*intelligent transportation systems – ITS*") tem integrado suas pesquisas mais desenvolvidas em soluções de controle de movimentos automáticos, dentro de um sistema chamado inteligência do veículo. Estes protótipos de veículos foram submetidos a vários testes em ambientes estruturados (trilha fechada). Em poucos casos estes testes foram feitos em estradas públicas com condições de tráfico real como são os casos seguintes:

- O protótipo *VaMP*, desenvolvido pela Universidade de Bundeswehr München (*UBM*), foi experimentado numa estrada de Münich (Alemanha) até Odense (Dinamarca) em 1995. O objetivo foi testar a capacidade, confiabilidade e desempenho do veículo autônomo em uma estrada. As tarefas automáticas que foram desempenhadas são: capacidade de manter-se na estrada seguindo as linhas e sinais, controle longitudinal, desvio de obstáculos, e manobras de mudança de estrada.
  O módulo de detecção de obstáculos e seguimento de linhas e sinais é baseado num processo de reconhecimento. Neste módulo foram implementados dois sub-módulos, o módulo de detecção e seguimento de um só objeto (*single object detector and tracker – SODT*) e o módulo de detecção e seguimento de múltiplos objetos (*multiple object detector – MODT*). O funcionamento de ambos sistemas tem como base um sistema de visão, composto por duas câmeras (par estéreo).
  O módulo de detecção e seguimento da estrada (*Road Detection and Tracking – RDT*) estima um estado de vetores o qual descreve um modelo dinâmico do movimento do veículo e da forma da estrada. Esta estimativa é realizada através de um algoritmo de filtro de Kalman.
- O sistema *RALPH* (*Rapidly Adapting Lateral Position Handler*) foi testado usando uma plataforma de navegação implementada num veículo *Pontiac*. O sistema de controle e navegação do veículo denominado *NavLab* 5, desenvolvido pela Universidade Carnegie Mellon, foi usado numa viagem chamada "*No Hands Across America*" desde Pittsburgh (PA) até San Diego (CA) em 1995. O *RALPH* é o cérebro do *Navlab* 5 baseado num sistema de visão composto por um par estéreo de câmeras.

Este veículo foi equipado com um computador portátil, câmeras, um receptor GPS, um sistema de detecção de obstáculos e outros equipamentos adicionais.

- O veículo experimental *ARGO*, ilustrado na Figura 2.2, foi testado usando o sistema *GOLD* (*Generic Obstacle and Lane Detection*) por quase 2000 km percorridos na Itália em 1998. Este sistema foi desenvolvido no Departamento da Engenharia da Informação da Universidade de Parma, Itália (*Dipartimento di Ingegneria dell'Informazione of the Universita di Parma*). O veículo usado, *Lancia Thema*, foi equipado com um sistema baseado em visão. A captura de imagens e realizada através de duas câmeras (par estéreo) as quais permitem extrair informação tanto da estrada como do ambiente por onde se movimenta o veículo. Através deste sistema são detectados e localizados os obstáculos ao longo da estrada, en quanto o processamento da imagem capturada permite extrair a geometria da estrada que fica na frente do veículo. Além disso o veículo está equipado com uma placa de entradas e saídas (I/O) usada para fazer aquisição de dados de velocidade.

Figura 2.2 – Veículo experimental ARGO (modificado – Broggi, 1999)

O veículo *ARGO* tem capacidade autônoma no controle do movimento e direção do veículo. O resultado do processamento das imagens capturadas, usada na detecção da posição dos obstáculos da geometria da estrada, é usado para controlar o atuador do sistema de movimento e direção da roda (*steering wheel*) quando o veiculo está-se movimentando (Bertozzi, et al., 2000).

Outro trabalho mais recente desenvolvido neste campo é o *IVVI* (*Intelligent Vehicle base on Visual Information*). Este veículo, como se mostra na Figura 2.3, é uma plataforma de pesquisa para a implementação de sistemas, baseados em visão por computador, que servem de ajuda no controle do veículo. Basicamente foram desenvolvidos sistemas de detecção de sinais de trânsito, outros veículos, pedestres e limites de estrada (Collado, et al., 2003).

Figura 2.3 – Veículo Experimental IVVI (modificado – Collado, 2003)

2.6 – SISTEMAS BASEADOS EM VISÃO EM VEÍCULOS DESENVOLVIDOS NO JAPÃO

Nesta seção serão apresentados os primeiros trabalhos desenvolvidos no Japão em veículos inteligentes e particularmente três deles descritos por (Tsugawa, 1994).

O primeiro dos veículos inteligentes foi desenvolvido no ano 1970, com um sistema de detecção de obstáculos baseado em um sistema de visão e um sistema de navegação que indica constantemente a posição do veículo no ambiente, através da constante estimativa das suas coordenadas cartesianas (*dead reckoning*). As velocidades alcançadas por este veículo foram de até 10 km/h .

O segundo é um veículo com um sistema pessoal (*Personal Vehicle System – PVS*) desenvolvido no ano 1980 baseado num sistema de visão. O sistema de visão captura as marcas da estrada (linhas centrais e laterais) por onde o veiculo está sendo guiado. O sistema desenvolvido para evitar obstáculos também foi baseado num sistema de visão usando um par estéreo de câmeras. As velocidades alcançadas por este veículo foram entre 10-30 km/h

O terceiro é um veículo automático para rodovias (*Automated Highway Vehicle System – AHVS*). Este sistema, também baseado em visão, é composto por uma câmera de TV para o seguimento de linhas na rodovia, controlado por um controlador *PD*. As velocidades alcançadas por este veículo foram de até 50 km/h.

2.7 – TRABALHOS DESENVOLVIDOS USANDO SISTEMAS BASEADOS EM VISÃO PARA NAVEGAÇÃO ROBÓTICA

Um dos trabalhos desenvolvidos para navegação robótica, usando como sistema base um sistema de visão, foi realizado em 1998 por o grupo de pesquisadores em robótica da Universidade de Oxford e apresentado numa tese de doutorado por Andrew John Davison (1998). O objetivo deste trabalho foi o desenvolvimento de um sistema de navegação robótica, baseado em visão, para realizar tarefas de localização e mapeamento, operando num ambiente desconhecido, usando sinais e marcas visuais (*landmarks*).

Outro trabalho importante baseado num sistema de visão, foi realizado em *INRIA* – Francia. Este sistema faz uma análise da seqüência de imagens capturas por uma câmera em movimento e tem como objetivo determinar a forma do ambiente por onde o robô se está movimentando (Faugeras, et al., 1995).

O projeto de um *AGV* desenvolvido também pela Universidade de Oxford, é outro trabalho importante que envolve diferentes aspectos da navegação robótica. O sistema desenvolvido usa vários tipos de sensores e métodos, que dão uma autonomia ao robô para realizar tarefas principalmente em aplicações industriais. Um dos aspectos chaves desenvolvidos neste projeto é a fusão sensorial a qual combina diferentes tipos de sensores (ultra-som, visão, laser, etc) que são parte do equipamento do robô para realizar a tarefa de navegação (Cameron, 1994)

Outra abordagem diferente realizada em navegação e robótica foi desenvolvida por Brooks (1987) no *MIT*. Brooks chamou este sistema de "Inteligência sem representação" do inglês "*Intelligence Without Representation*" o qual consiste numa coleção de simples comportamentos, interagindo entre eles, com um computador central de tomada total de decisões. Estes comportamentos são chamados de camadas (*layer*) e são divididos basicamente em três partes : (1) exploração (sistema de visão), (2) evitar obstáculos (ultra-som e

três partes : (1) exploração (sistema de visão), (2) evitar obstáculos (ultra-som e mapeamento) e (3) movimentação (Brooks, 1991).

Outros trabalhos desenvolvidos são:

- Detecção de linhas e obstáculos baseados em um algoritmo rápido de transformação de perspectiva "*fast inverse perspective mapping algorithm (FIPMA)*", (Jiang, et al., 2000).
- Um estúdio de reconhecimento de linhas de estradas e movimentos de veículos usando sistemas de visão (Park, et al., 2003).
- Detecção geométrica de linhas para guiamento de veículos inteligentes (Wong, et al., 1999)
- Reconhecimento de linhas na estrada em tempo real usando lógica fuzzy para sistemas de visão de veículos guiados automaticamente (*AGV*), (Kuang, et al., 2004)

Todos os métodos utilizados nos diferentes trabalhos mencionados neste capítulo, na navegação, controle, detecção e localização de veículos, podem ser divididos em dois grupos: 1) métodos de geração de hipóteses (*HG*) e 2) métodos de verificação de hipóteses (*HV*) (Sun, Z., et al., 2005). Estes métodos, que são utilizados nos veículos autônomos para estrada, são utilizados também na implementação de sistemas de navegação robótica e em veículos guiados automaticamente para aplicações industriais (*AGV's*).

3 - SISTEMAS FLEXÍVEIS DE MANUFATURA – CONCEITOS BÁSICOS

A palavra “sistema” significa um conjunto de elementos interligados, destinados a uma determinada função. No nosso caso, essa função é a produção de bens.
Costuma-se utilizar a palavra “manufatura” para significar produção, embora em seu sentido original – “fazer à mão” – a palavra não represente a realidade atual, em que cada vez mais as máquinas substituem a habilidade manual do homem.

Assim, a expressão “sistema de manufatura” não é nova. A novidade está no adjetivo “flexível”. Essa característica foi se incorporando aos sistemas de produção à medida que a presença dos computadores nesses sistemas se tornava mais freqüente.

Embora não haja consenso entre os vários autores quanto à origem do primeiro sistema flexível de manufatura, alguns consideram a indústria inglesa de máquinas-ferramenta “*Mollins*” como sendo a primeira a implantar, em 1968, um sistema desse tipo. Ele teria sido construído para fabricar uma grande variedade de componentes e pode opera sem a presença do homem por longos períodos.

Desde a década de 60, os sistemas flexíveis de manufatura tornaram-se cada vez mais sofisticados. Os pioneiros no estudo dos métodos e processos de produção criaram, no início do século, algumas formas de representar os sistema de produção. Uma dessas formas é a dos chamados fluxogramas de produção. Com símbolos representavam-se as várias fases pelas quais passava o material ao ser processado. Assim, podiam-se indicar as operações, inspeções, transportes, armazenamentos e tempos de espera do material em processo. Em seguida, tentava-se reduzir ou eliminar as atividades que apenas aumentavam o custo do produto. Os fluxogramas foram substituídos por formas mais modernas de representação, muitas delas realizadas com o auxílio do próprio computador.

3.1 - DEFINIÇÃO DE SISTEMA FLEXÍVEL DE MANUFATURA (FMS)

Segundo Groover (2003) um *FMS* é definido como um dos tipos de células de máquinas usados para implementar uma tecnologia de grupo (*TG*). Estas células como parte do *TG* são bem automatizadas e tecnologicamente sofisticadas. No esquema de classificação pro-

posto pelo mesmo autor para sistemas de manufatura, um *FMS* possui tipicamente múltiplas estações automatizadas interagindo entre elas. Um *FMS* altamente automatizado inclui vários conceitos e tecnologias, como: máquinas CNC, controle computadorizado distribuído, transportador de material (*AGV*) e armazenamento automatizado e tecnologia de grupo.

O conceito de *FMS* foi originado in Grã-Bretanha em 1960. A primeira instalação de uma *FMS* foi feita nos Estados Unidos em 1967. Estes sistemas criados inicialmente executaram manufaturas de famílias de partes usando máquinas ferramentas de Controle Numérico (*NC*)

Um sistema flexível de manufatura (*FMS*) é formado por máquinas-ferramenta com comando numérico computadorizado (*CNC*), interconectadas para transferência de informações (por um sistema de rede de comunicações) e transferência de materiais (por um sistema de manipulação, troca e transporte de peças e ferramentas como: esteiras, veículos transportadores, etc.). Um *FMS* se distingue de outras formas de manufatura automatizada por considerar:

- A diversidade de produtos que se deseja produzir (flexibilidade do produto);
- As características adaptativas das máquinas (flexibilidade do produto);
- As características adaptativas das máquinas (flexibilidade dos equipamentos); e
- As propriedades de similaridade dos processos (flexibilidade do processo).

Consideram-se também as implicações nas relações de *custo/benefício* e *produtividade/qualidade* do sistema. Tal diversidade exige do processo uma grande versatilidade para lidar com diferentes combinações operacionais e funcionais de máquinas ferramentas e seqüências de processos.

3.1.1 - O Termino “Flexível” aplicado a um FMS

Segundo Groover (2003) , há três características fundamentais que deve ter um sistema de manufatura para que possa ser flexível:

1. A habilidade de identificar e distinguir as diferentes partes ou tipos de produtos processados pelo sistema.
2. Rapidez de troca das instruções de operação
3. Rapidez na execução e mudança dos "*setups*" das partes físicas do sistema.

O termo flexibilidade pode ser aplicado tanto a sistemas manuais como automatizados. Num sistema manual o operário faz que o sistema seja flexível.

Para poder desenvolver o conceito de flexibilidade num sistema de manufatura automatizado, pode-se considerar uma célula de máquinas formada pelos seguintes elementos: Duas máquinas-ferramenta CNC as quais são carregadas e descarregas por um robô industrial e uma esteira re-circulante de transporte de partes que alimenta o robô como se mostra na Figura 3.1. Periodicamente um operário deve descarregar completamente as partes da esteira e substituí-las com novas partes para serem trabalhadas. Por definição esta é uma célula de manufatura automatizada, mas é uma célula flexível de manufatura? Alguns poderiam argumentar sim, é flexível, já que a célula está conformada de máquinas ferramentas *CNC* e estas máquinas *CNC* são flexíveis porque elas podem-se programar para fabricar diferentes tipos de partes previamente configuradas. Contudo se a célula só opera como um sistema de produção em grupo (*batch mode*) na qual só um tipo de parte é produzida em ambas máquinas e em lotes grandes de unidades (dezenas ou centenas) então isso faz que não qualifique como uma manufatura flexível.

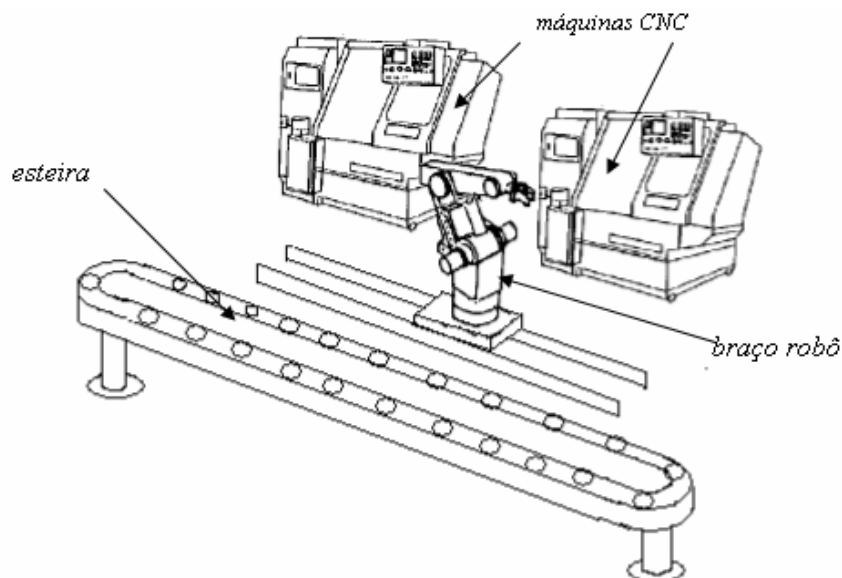



Figura 3.1 – Célula de manufatura automatizada, com duas máquinas CNC, um robô e uma esteira re-circulante de transporte de partes.

Já um sistema de manufatura automatizado para qualificar como flexível deve satisfazer os seguintes critérios:

1. **Teste de variedade de partes**: Pode o sistema processar diferentes tipos de partes não trabalhando num sistema de produção em grupo ? ("*nonbatch mode*")
2. **Teste de mudanças de programação:** Pode o sistema aceitar mudanças na programação da produção e mudanças nas quantidades de produção nos diferentes tipos de partes ?
3. **Teste de identificação e recuperação de erros:** Pode o sistema recuperar eficientemente erros ou más funções de operação e colapsos dos equipamentos que conformam a célula ?
4. **Teste de partes novas:** Pode o sistema aceitar novas partes projetadas dentro os diferentes produtos com relativa facilidade ?

Se a resposta para todas estas perguntas é "sim", então o sistema pode ser considerado flexível. Os mais importantes critérios são o (1) e o (2). Já os critérios (3) e (4) são menos determinantes sem deixar de ser importantes e podem ser implementados em vários níveis.

Célula Flexível de Manufatura

Uma Célula Flexível de Manufatura pode estar configurada por dois ou três estações de trabalho (máquinas *CNC* ou centros de torneamento), mais um sistema de manipulação de partes do inglês "*part handling system*" como se mostra na Figura 3.2. O sistema de manipulação é conectado a uma estação de carga e descarga de partes. Adicionalmente ao sistema de manipulação usualmente inclui um sistema de armazenamento de capacidade limitada.

Sistema Flexível de Manufatura

Já um *FMS* tem quatro ou mais estações de trabalho de processamento (máquinas *CNC* ou centros de torneamento) conectados mecanicamente por um sistema de manipulação de partes comum e eletronicamente por um sistema computadorizado distribuído. A reunião de várias células de manufatura dá origem ao chamado sistema de manufatura.

Uma importante diferença entre um *FMS* e uma *FMC* é o número de máquinas: Um *FMS* tem quatro ou mais. Uma segunda diferença é que um *FMS* tem geralmente estações de trabalho como suporte da produção mas não participam diretamente nela. Estas estações podem ser armazém de partes, estação de limpeza, estação de medição de coordenadas, etc. Uma terceira diferença é que o sistema de controle computadorizado de um *FMS* é, geralmente muito mais complexo e sofisticado. O esquema apresentado na Figura 3.2 mostra a variação dos fatores de investimento, nível de produção e volume ao ano versus o número de maquinas instaladas e o tipo de instalação que deveria ser feita (*FMC*, *FMS*).

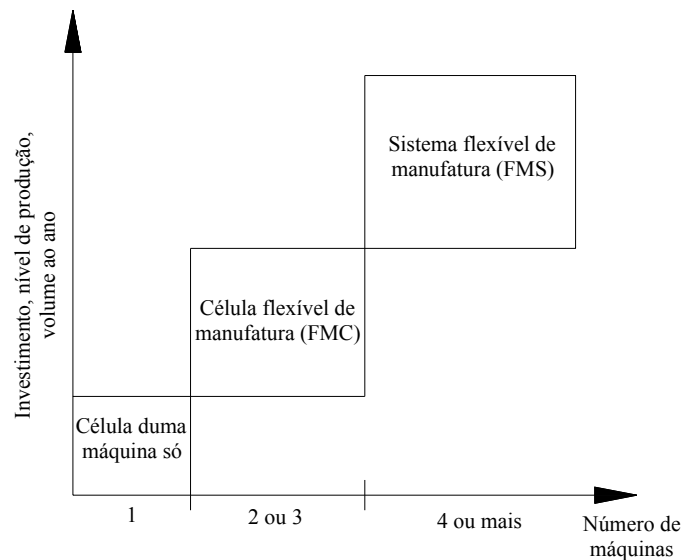



Figura 3.2 - Características das três categorias numa *FMC* e uma *FMS* (modificado - Groover, 2003)

3.2 - COMPONENTES DE UMA FMS

Como já foi indicado um *FMS* é formado basicamente pelos seguintes componentes:

i. Estações de trabalho do inglês "*workstations*"
ii. Manipulador de material e sistema de armazenamento
iii. Integração e comunicação (Um *FMS* é altamente automatizado)
iv Recursos humanos

i. Estações de trabalho (*workstations*)

- O equipamento montado usado num *FMS* depende do tipo de trabalho que será efetuado pelo sistema. Embora nos sistemas projetados sejam na maioria dos casos para trabalhos de manufatura e os principais tipos de estações de processamento se-

jam as máquinas ferramenta CNC, o conceito de *FMS* é aplicado em vários outros processos. Os tipos de estações comumente utilizados num *FMS* são: estações de carga e descarga (*load / unload stations*), estações de máquinas (workstations - CNC, *mill-turn centers*, etc). Entre outras estações, dependendo do processo de manufatura estão: estações de montagem, estações de controle da qualidade, limpeza, refrigeração, etc.

ii) Manipulador de material e sistema de armazenamento (*Material Handling and Storage System*)

Nesta seção serão apresentadas, de uma forma geral, as funções do sistema de manipulação de material e de armazenamento, assim como também serão apresentados os diferentes equipamentos que compõem este sistema tipicamente usado em um *FMS*.

As principais características dos sistemas manipuladores de material são:

- Movimentos independentes peças (*workparts*) entre estações
- Manipulação duma grande variedade de peças
- Armazenamento temporário
- Acesso conveniente na carga e descarga de peças
- Compatibilidade com o sistema de controle computadorizado.

Há vários tipos de máquinas, controladas por computador, destinadas a transportar materiais. Entre elas, destacam-se os *AGV's* e os *RGV's*. Esses nomes, são siglas de termos em inglês "*Automatically Guided Vehicle*", ou seja, Veículo Guiado Automaticamente; e "*Rail Guided Vehicle*", ou seja, Veículo Guiado por Trilho. A Figura 3.3 (a, b, c) ilustra alguns dos equipamentos comumente utilizados no sistema de transporte e manuseio de material na industria moderna. Já os sistemas *AS/RS* do inglês "*Automatic Storage / Retrieval System*" são outros dispositivos com maior aplicação em empresas industriais ou de distribuição e são sistemas fundamentais de suporte ao manuseio e armazenagem automáticos de materiais em sistemas de manufatura automatizada, e em cadeias de distribuição de produtos automatizadas. A Figura 3.3 (d) mostra uns exemplos destes sistemas automáticos de armazenamento.

a) b) c)

d)

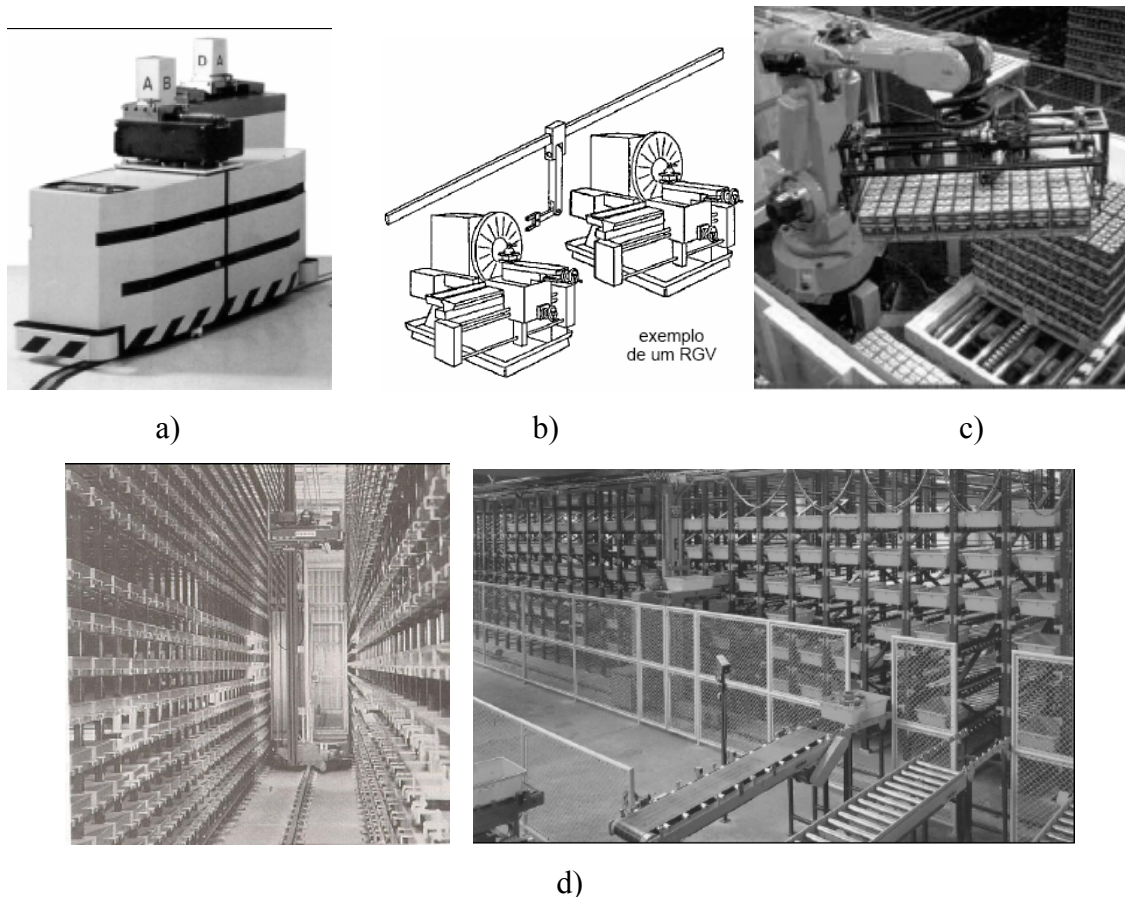



Figura 3.3 – Exemplos de manipuladores e armazenes na industria moderna (a) *AGV*, (b) *RGV*, (c) braço robô e (d) *AS/RS* (modificado – Leitão, 2005)

iii. Integração e Comunicação

Para que os equipamentos de produção trabalhem de forma cooperativa, é necessário que estejam integrados, ou seja, conectados a um controle central, encarregado de comandá-los de forma harmônica. Este sistema está interconectado com as estações de trabalho, o sistema manipulador de material e outros componentes de hardware do *FMS*. Um sistema de controle típico de um *FMS* é formado por um computador central e outros computadores controlando cada uma das máquinas independentemente. O computador central coordena as atividades dos componentes durante a operação do *FMS*. As funções do sistema de controle podem ser agrupadas nas seguintes categorias:

- Controle das estações de trabalho
- Distribuição das instruções de controle nas estações de trabalho.
- Controle e administração da produção .

- Controle do trânsito dos sistemas manipuladores durante a movimentação das partes entre as estações de trabalho.
- Controle do transporte dos sistemas secundários existentes em cada uma das estações.
- Monitoramento do estado das peças de trabalho.
- Controle e administração de ferramentas.
- Desempenho, monitoramento e informações.
- Diagnósticos, indicar as prováveis fontes que originam um problema no *FMS*

O controle central troca informações com os controladores dos equipamentos de produção por meio de uma rede de comunicação. A Figura 3.4 mostra um esquema de uma rede de comunicação comumente utilizada nas implementação de *FMS*'s.

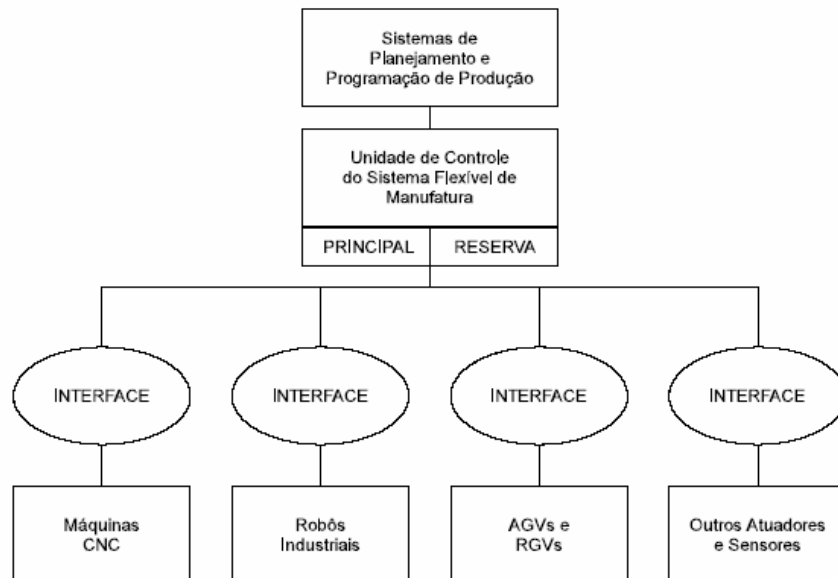



Figura 3.4 – Esquema de uma rede de comunicação (modificado - Leitão, 2004)

iv. Recursos Humanos

Um componente adicional em um *FMS* é a mão de obra. Humanos são necessários para administrar as operações do *FMS*. As funções comuns realizadas são: (1) carregar material ou peças no sistema, (2) descarregar partes terminadas ou montadas do sistema, (3) troca e seleção de ferramentas, (4) manutenção e reparação dos equipamentos, (5) programação das partes no sistema das máquinas CNC (6) programação e operação do sistema computadorizado e (7) Administração total do sistema.

3.3 – APLICAÇÕES, BENEFICIOS E LIMITAÇÕES DE UMA FMS

3.3.1 - Aplicações de uma FMS

O conceito de automação flexível é aplicável a uma grande variedade de operações ou processos de manufatura. Algumas das aplicações onde é comumente utilizado um *FMS* são: operações de manufatura com máquinas ferramentas (a mais comum aplicação), montagens de partes, inspeção, conformação de metais, etc.

3.3.2 - Benefícios de uma FMS

Entre os principais benefícios que se obtém ao implementar um *FMS* estão:

- Incremento da utilização das máquinas
- Poucas máquinas requeridas
- Redução do espaço do chão de fabrica
- Capacidade de mudança alta
- Redução de inventários
- Baixos tempos mortos de produção
- Alta produtividade e qualidade do produto ótima
- Tempos grandes de produção sem intervenção do operário.

3.3.3 - Limitações de uma FMS

As limitações mais relevantes são:

- Não é uma panacéia para solucionar todos os problemas de produtividade e de lucro.
- Custo alto de produção não justificado pode ser negativo
- Integração com sistemas existentes podem não dar certo

4 – SISTEMAS MANIPULADORES DE MATERIAL

Um manipulador de material é definido por "*The Material Handling Industry of América*", como o elemento responsável pelo movimento, armazenamento, proteção e controle dos materiais por todo o processo de manufatura e de distribuição, incluindo o consumo e disposição deles.

A manipulação do material deve ser feita com uma segurança alta, eficiente, custo baixo, precisão, tempos exatos e seqüenciados e sem estragar o material. O custo do manipulador de material é uma parte importante no custo total de produção; em média ao redor de (20 – 25%) do custo total de manufatura nos Estados Unidos (Eastman, 1987). O custo, em proporção, dos manipuladores de material, varia dependendo do tipo de produção e grau de automação. Além dos sistemas de manipulação, transporte e armazenamento de material mencionados anteriormente como são os *AGV's* e AS/RS, existem outros tipos de manipuladores de material como, robôs industriais, "*pallets*" transportador em máquinas *CNC*, esteiras usadas numa linha de montagem manual, mecanismos de transferência e alimentador de partes em linhas automatizadas.

4.1 – CONSIDERAÇÕES NO PROJETO DE SISTEMAS MANIPULADORES DE MATERIAL

Os equipamentos usados como manipuladores de material são usualmente montados dentro de um sistema. O sistema deve ser especificado e configurado para satisfazer os requerimentos duma determinada aplicação. Já o projeto do sistema depende dos materiais que serão manipulados, quantidades e distancias que serão movimentados, e outros fatores.

Os materiais são classificados em categorias por suas características físicas: estado físico, tamanho, peso, forma, segurança e risco, etc.

4.1.1 - Fluxo , Rotas e Programação da Produção ("*Scheduling*")

Outros fatores importantes que devem ser considerados na análise do sistema e assim determinar que tipo de equipamento é o mais apropriado, são: quantidade e fluidez do material a ser movimentado, controle de rotas e programação seqüenciada dos movimentos.

A Figura 4.1 mostra, como uma referencia geral, um guia de como selecionar os equipamentos manipuladores de material para algumas aplicações características, envolvendo especificamente fluidez e distancia movimentada (Groover, 2003).

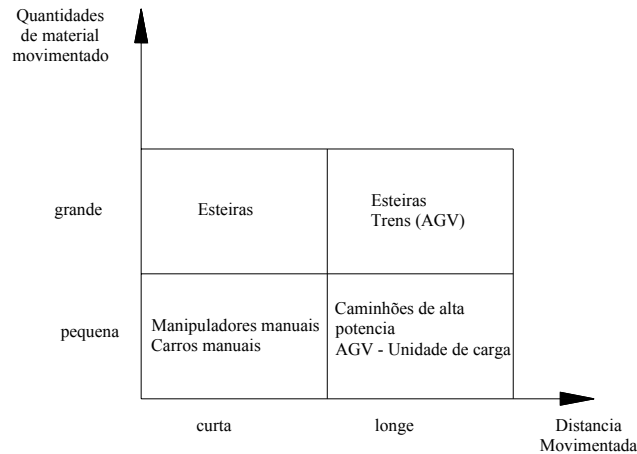



Figura 4.1 - Esquema para selecionar os equipamentos manipuladores de materiais (modificado - Groover, 2003)

4.1.2 - Os 10 princípios dos Manipuladores de Material

Segundo Groover (2003), os sistemas manipuladores de material se fundamentam em 10 princípios básicos: 1) Planejamento, 2) Padronização dos Equipamentos, 3) Trabalho ótimo, 4) Ergonometria, 5) Unidade de carga adequada, 6) Utilização de espaço, 7) Sistema, 8) Automação, 9) Impacto no Ambiente, e 10) Custo e tempo de vida.

4.2 - SISTEMAS DE TRANSPORTE DE MATERIAL

Os principais tipos de transporte de materiais comumente usados para movimentar peças ou outros materiais nas áreas de processamento e armazenamento são: (1) carros industriais de alta potência, (2) veículos guiados automaticamente (*AGV*), (3) veículos guiados através de trilhos, (4) esteiras e (5) talhas e guindastes. A Tabela 3.1 mostra um resumo da classificação dos equipamentos comumente usados nos sistemas de transporte e manipulação de material assinalando suas características econômicas e aplicações típicas.

Tabela 4.1 - Resumo de características e aplicações dos equipamentos usados nos sistemas de transporte e manipulação de material (modificado – Groover, 2003)

Equipamento – Manipuladores de material	***Características***	***Aplicações típicas***
Carros industriais manuais	Baixo custo Baixo índice de entregas /hora	Movimento de cargas leves (1-2ton)
Carros industriais potentes	Custo médio	Movimento de pallets e containers na fábrica
Sistemas de veículos guiados automaticamente	Alto custo Uso de baterias Rotas flexíveis Caminhos não estruturados	Movimento de pallets e containers na fábrica e armazens. Movimentos durante o processo de produção (baixa e medía) através de rotas variáveis.
Veículos guiados através de trilhos	Alto custo Caminhos ou rotas flexíveis Tipos terrestres ou aéreos	Movimentos de grandes quantidades de itens através de rotas fixas na fábrica
Esteiras de alta potência	Grande variedade nos equipamentos	Movimento de produtos através de linhas de montagem manual
Talha e guindastes	Capacidades maiores de 10 tons	Movimentos grandes de itens muito pesados (>10 tons)

4.2.1 - Veículos Guiados Automaticamente (AGV)

Um *AGV* é um dispositivo móvel utilizado no transporte automático de materiais em ambientes de manufatura, concebido para receber e executar instruções, seguir um caminho ou trajetória, e aceitar e distribuir materiais. As instruções para um *AGV* indicam para onde o

veículo se deve dirigir, como deve chegar ao destino e que deve fazer quando chegar ao destino (Rocha, 1998). Os *AGV's* foram introduzidos na industria manufatureira in 1955 e atualmente mais de 20,000 *AGV's* são utilizados em aplicações industriais. (Miller, 1987).

Os veículos guiados automaticamente são usados para transporte interno e externo de materiais. Tradicionalmente os AGVs foram mais usados em sistemas de manufatura; atualmente os AGVs são utilizados para tarefas de transporte repetitivas e em outras áreas tais como armazenagem, sistemas de transporte externo (subsolo), etc.

O uso dos AGVs tem crescido enormemente desde sua introdução. O número de áreas de aplicação e variação nos tipos têm crescido significativamente. Armazéns e centros com muitas intersecções são exemplos de áreas distribuídas.

Os *AGV's* são utilizados nestas áreas para o transporte interno, por exemplo: "*pallets*" entre diferentes áreas, tais como entrada de materiais, armazenamento, classificação e embarque.

O sistema utilizado num veículo guiado automaticamente é um sistema manipulador de material que trabalha independentemente, auto propulsado, guiado através de rotas ou caminhos definidos. A potência necessária para movimentar estes veículos é proporcionada por um sistema de baterias transportadas no mesmo *AGV* as quais permitem um tempo longo de funcionamento (8 – 16 horas; *AGV's* industriais). Um *AGV* é apropriado para trabalhar em industrias nas quais diferentes materiais são movimentados de vários pontos de carga a vários pontos de descarga. Os avanços tecnológicos dos *AGV's* atualmente são desenvolvidos com tecnologias eletrônica e computacional. Assim *AGV*'s industriais possuem sistemas de navegação sensorial, controle inteligente, gerenciamento total do sistema e sistema de segurança.

4.2.2 - Características dos *AGV'S*

Os *AGV's* têm cinco características básicas:

1. Facilidade de Direção e localização ("*guidance*")
2. Decisão ou escolha de rota ("*routing*")

3. Gerenciamento e Controle do Trânsito dos *AGV's* ("*traffic management*")
4. Transporte e transferência de carga ("*transfer load*")
5. Sistema de administração e controle das funções dos *AGV's* ("*System management*")

4.3 - ROBÔS INDUSTRIAIS

O estudo e desenvolvimento dos mecanismos de robô se originaram pelos anos quarenta, quando foram projetados e fabricados no *Oak Ridge* e *Argonne National Laboratories* manipuladores mestre – escravo para o manuseio de materiais radioativos. O primeiro robô industrial comercial controlado por um computador foi introduzido aos finais dos anos cinqüenta pela *Unimation*, Inc., sendo que nos quinze anos seguintes uma serie de dispositivos industriais e experimentais foram desenvolvidos (Fu, et al., 1990).

Um robô industrial é uma máquina programável que possui certas características antropomórficas. A característica antropomórfica mais ressaltante é seu braço mecânico que é usado para realizar uma grande variedade de tarefas industriais. Outras características parecidas às humanas são a capacidade do robô de responder a entradas de sinais de sensores desde outras máquinas e realizar as tarefas correspondentes. A tecnologia desenvolvida nestes robôs envolve um controle coordenado de eixos múltiplos com um computador digital como controlador central do robô (Groover, 2003).

4.3.1 - Aplicações dos Robôs Industriais

As características gerais pelas quais os robôs devem substituir a função humana em situações de trabalhos industriais são as seguintes:

- Ambientes de trabalho perigosos para humanos
- Ciclos repetitivos de trabalho
- Tarefas de manuseio de materiais dificultosas para humanos
- Versatilidade na troca e execução de tarefas
- Versatilidade no posicionamento e orientação das partes de trabalho.
- Fácil reprogramação do robô para a execução de novas tarefas.

4.3.1.1 - Aplicações dos Robôs Industriais como Manipulador de Materiais

As aplicações executadas por um robô industrial como manipulador de materiais são basicamente as relacionadas ao movimento de partes ou materiais desde um lugar a outro. Para que o traslado destas partes ou materiais possa ser feito o robô industrial é dotado de uma garra que trabalha na extremidade do braço do robô. A garra deve ser projetada para segurar uma parte ou partes específicas que serão movimentadas na tarefa desenvolvida pelo robô. As aplicações fundamentais do robô trabalhando como manipulador de materiais ou partes são as seguintes: (1) Transferência de material e (2) Carga e descarga de máquinas. Em quase todas as aplicações do robô como manipulador de materiais tanto as posições como as orientações das partes ou materiais devem ser conhecidas pelo robô através de uma adequada programação das tarefas específicas.

4.3.1.2 - Operações de Processamento

Estas aplicações são aquelas nas quais o robô executa uma operação de processamento nas peças. A característica principal; que distingue os robôs neste tipo de aplicação, é que o robô está equipado com algum tipo de ferramenta na extremidade do braço. Através da manipulação da ferramenta o robô cumpre um ciclo de trabalho durante o processamento das partes que estão sendo trabalhadas. Entre as aplicações típicas destas operações de processamento, que são executadas pelos robôs industriais, estão soldagem por pontos, soldagem por arco continuo, pintura com spray, etc (Groover, 2003).

4.3.1.3 - Montagem e Inspeção

As operações de montagem e inspeção executadas pelos robôs industriais geralmente são uma combinação das aplicações de manipulador de material e processamento.
Montagem e inspeção são geralmente atividades fortemente intensivas, altamente repetitivas e tediosas. São estas as razões pelas quais os robôs são máquinas ideais para realizar estas aplicações.

4.4 - ROBÓTICA INDUSTRIAL VS. ROBÓTICA MÓVEL

É necessário definir exatamente as características específicas que definem um manipulador e as que definem um robô móvel (Leitão, et al., 2005). Tanto os manipuladores como os robôs móveis estão dentro da categoria de robôs; dependendo das características físicas de cada um deles podem realizar determinadas tarefas dentro de um *FMS*. A Tabela 4.2 mostra as características mais importantes que definem cada um deles.

Tabela 4.2 – Características principais dos manipuladores e robôs móveis (modificado - Leitão, 2005).

Robótica Industrial	**Robótica Móvel**
Robôs	
Manipuladores	**Robôs Móveis**
• Base fixa – capacidade de mobilidade limitada	• Base móvel
• Manipuladores multi-aplicação multi-funcional e re-programável	• Possui capacidade de mobilidade
• Realização de tarefas repetitivas normalmente atribuídas a humanos	• Realiza as suas próprias decisões utilizando o *feedback* que recebe do seu ambiente

5 - ESTADO DA ARTE DA ROBÓTICA MÓVEL APLICADA EM AGVs.

Com o desenvolvimento de tecnologias de navegação para veículos autônomos e o aumento da capacidade de processamento dos computadores, apareceram os primeiros robôs moveis industriais, cujo campo de aplicação é muito amplo. Segundo Rocha (2001) um robô móvel procura conjugar num só dispositivo de automação a mobilidade de um veículo autônomo e a capacidade de manuseamento e manipulação dos robôs. Neste domínio, o AGV (*Automatic Guided Vehicle*) e o AS/RS (*Automatic Storage / Retrieval System*) são os dispositivos com maior aplicação em empresas industriais ou de distribuição.

Um sistema flexível de manufatura, como já se definiu no capitulo anterior, esta formado por um conjunto de máquinas e por um sistema de transporte e manuseamento de materiais, ligados e controlados através de uma rede de computadores. Antes do aparecimento da robótica móvel, o transporte e manuseio de materiais eram realizados com uma forte intervenção humana (ex-veículos guiados manualmente). Com o desenvolvimento de tecnologias de navegação para veículos autônomos e o aumento da capacidade de processamento dos computadores, apareceram os primeiros robôs móveis industriais, com o objetivo de reduzir a intervenção humana nos sistemas flexíveis de manufatura (década 80).

A utilização de robôs móveis em um sistema de manufatura permite:

- Aumentar o grau de automação e flexibilidade, facilitando a integração total e controle otimizado do sistema, através de uma rede de computadores.
- Otimizar o fluxo de materiais, através de um correto escalonamento de tarefas, contribuindo para uma melhoria significativa da produtividade total do sistema.
- Eliminar a presença humana em ambientes potencialmente agressivos e perigosos para a saúde.

Um robô móvel procura conjugar num só dispositivo a mobilidade de um veículo autônomo e a capacidade de manuseio e manipulação dos robôs convencionais (braços robôs, garfos telescópicos, etc.).

O aparecimento de robôs móveis trouxe consigo reconhecidas vantagens, mas também inúmeros desafios tecnológicos, quais sejam:

- Alimentação a partir de baterias recarregáveis, que procura dar ao robô a maior duração de tempo de trabalho autônomo.
- Sistemas de navegação automáticos, eficientes, flexíveis, tolerantes a falhas e seguros.
- Controlo eficiente de frotas constituídas por vários robôs móveis, resolvendo problemas como o escalonamento otimizado, o encaminhamento e gestão do tráfego.

Assim o desenvolvimento de robôs móveis é uma tarefa fortemente interdisciplinar, envolvendo áreas tecnológicas tão diversas como: sensores e atuadores, eletrônica de potência, energia, projeto mecânico, cinemática, dinâmica, teoria de controle, escalonamento em tempo real, investigação operacional, sistemas de informação, telecomunicações, etc.
Em nível internacional, o campo de aplicação dos robôs móveis não se restringe a indústria, abrangendo também as áreas de logística (distribuição e armazenagem), exploração subaquática e oceanográfica, exploração planetária, bem como aplicações militares.

5.1 – TECNOLOGIA DESENVOLVIDA PARA GUIAR OS VEÍCULOS AUTOMATICAMENTE

O sistema de guiagem nos *AGV's* utiliza métodos pelos quais seus caminhos são definidos e controlados para seguirem os esses caminhos. Existem três tecnologias as quais são comumente usadas em sistemas comerciais dos *AGV's*: (1) cabo guiado, (2) rastreamento de caminhos desenhados não chão "*paint strips*" e (3) veículos autoguiados.

Dos três tipos de tecnologia desenvolvida mencionados, a utilizada nos veículos autoguiados é a mais moderna e sofisticada que engloba as técnicas mais recentes. A diferença dos outros dois métodos é que estes veículos operam sem caminhos contínuos definidos. Em vez disso ele usa uma combinação do sistema "*dead reckoning*" (estimativa de posição sem uma referencia externa) com "*beacons*" (sinais artificiais) localizados por toda a planta os quais podem ser identificados pelos sensores do veículo. "*dead reckoning*" se refere à capacidade do veículo de seguir uma rota dada na ausência de um caminho definido no

chão. O movimento do veículo ao longo da rota é afetado pelos dados processados do número de revoluções das rodas. O processamento dos dados é realizado por um computador instalado no mesmo veículo. É esperado que a exatidão do posicionamento do robô seja afetada quando as distancias percorridas são maiores. Assim, a localização do AGV deve ser periodicamente verificada, comparando as posições calculas com as conhecidas. As posições conhecidas são estabelecidas usando sinais "*beacons*" localizadas estrategicamente em toda a planta. Há vários tipos de sinais usados nestes sistemas de *AGV* comerciais.

Um dos sistemas usados é o de código de barras montados nos corredores da planta. Estes podem ser captados por um scanner de laser montado no veículo. Baseado nas posições dos sinais o computador instalado no veículo para processar as tarefas de navegação usa triangulação atualizando as posições calculadas pelo sistema "*dead reckoning*". Outros sistemas usam sinais magnéticos colocados não chão da planta ao longo dos caminhos. O sistema de estimativa de posição é usado para movimentar o veículo entre os sinais, a atual localização delas fornece os dados para atualizar o mapa de dados estimados por "*dead reckoning*". Estas novas capacidades desenvolvidas permitem ao veículo acessar a lugares os quais seriam inacessíveis através de sistemas de guias com cabos.

A principal vantagem da tecnologia usada nestes veículos em relação às que utilizam caminhos fixos (guias com cabos, caminhos com sinais) é a flexibilidade. A rede de caminhos pode ser completamente mudada dependendo dos requerimentos dos dados processados do sistema de navegação. A rede de caminhos pode ser expandida instalando novas sinais ou "*beacons*". Estas mudanças podem ser realizadas rapidamente sem maiores alterações das instalações da planta.

5.2 - SISTEMAS DE NAVEGAÇÃO COMUMENTE UTILIZADOS EM *AGV's*

Existem diversos métodos de navegação que permitem um *AGV* seguir um caminho fixo ou um caminho dinâmico. A determinação dos caminhos fixos ou dinâmicos depende dos custos de instalação, dos requisitos de flexibilidade e da necessidade ou não da futura ampliação do sistema. Os sistemas com caminhos fixos são menos dispendiosos, todavia inviabilizam a possibilidade de reagir a alterações no *layout* do trabalho sem interromper seu funcionamento normal, acarretando por isso em custos adicionais.

5.2.1 - Sistemas de Caminhos Fixos

- **Sistema Filoguiado:** O sistema filoguiado, segundo Rocha (2000), como se mostra na Figura 5.1, é um exemplo de um método de caminhos fixos, que é usado em grande escala, devido a sua robustez e simplicidade. Este sistema baseia-se no seguimento do campo magnético criado por condutores implantados no solo e percorrido por uma corrente elétrica sinusoidal. O campo magnético é detectado por antenas, que seguem a freqüência corresponde ao caminho a seguir. Este sistema tem sido largamente utilizado devido à sua simplicidade e robustez, mas tem a desvantagem de não permitir a re-configuração do layout, o que o torna muito pouco flexível e inadequado a indústrias do tipo *job shop*, onde existe a necessidade de re-configurar freqüentemente o *layout* de fábrica.

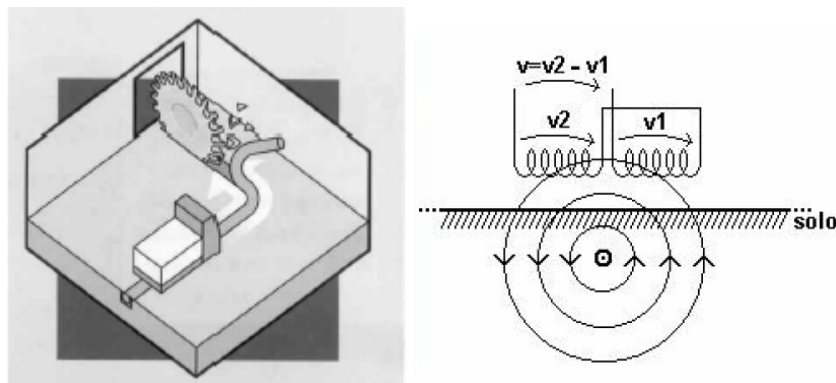



Figura 5.1 – Sistema filoguiado (Leitão, 2005)

- **Sistema baseado em Rastreamento de Linhas através de Visão Computacional:** A navegação realizada pelo veículo é realizada através da captura de informação visual obtida de marcas artificiais pré-existentes ("*landmarks*"), que são especificamente, linhas ou sinais colocados no ambiente onde trabalha o *AGV*. A informação obtida é eficientemente processada usando algoritmos de visão computacional (Beccari, et al., 1997). Este sistema é muitas vezes a solução preferencial devido a seu baixo custo de implementação e por ser uma boa alternativa para navegação de veículos em ambientes estruturados, como é o caso de uma célula flexível de manufatura.

A empresa Brasileira "AAT - Sistemas de Movimentação" explica o funcionamento do controle de direção ótico por câmera, através do sensor HG 73800, usado nos *AGV's* que eles fornecem. Este sensor é mostrado na Figura 5.2 (a) e serve para o guiamento ótico automático por trilha e identificação de marcas pré-existentes nos ambientes da fábrica.

O funcionamento do sensor consiste de uma câmera montada debaixo do veículo que vai registrando uma trilha ótica, como mostra a Figura 5.2 (b), O computador montado no próprio veículo avalia a informação da imagem e reconhece o desvio da trilha.

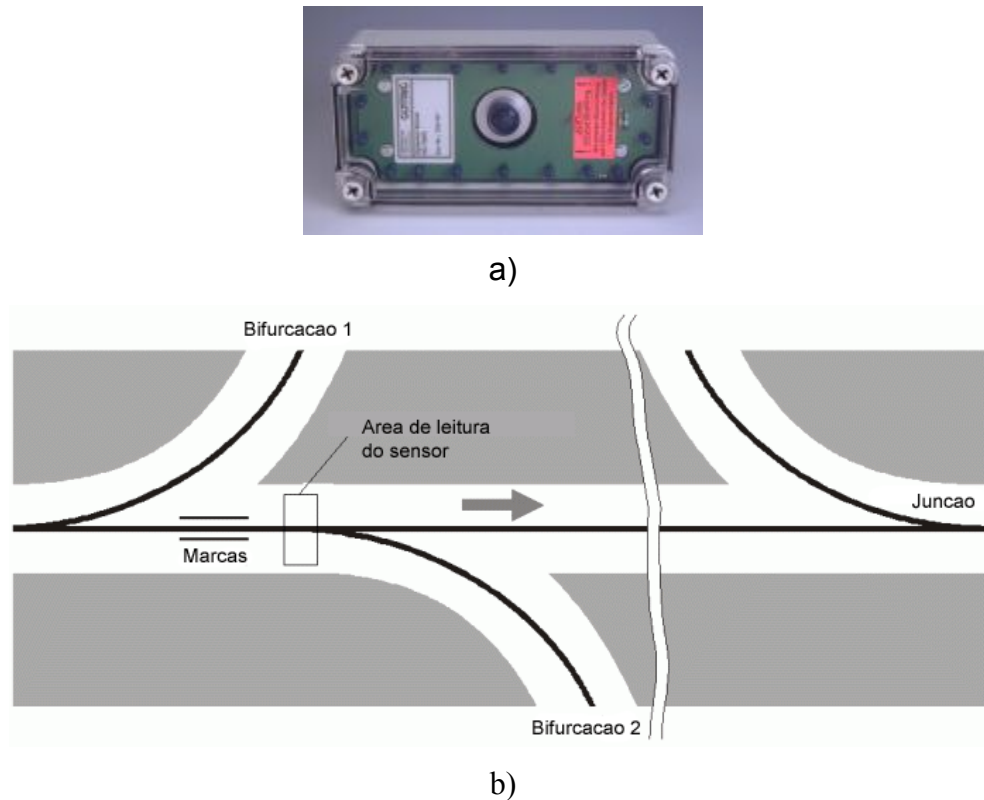



Figura 5.2 – Sensor de visão com câmera. (a) Sensor de guiamento e controle de direção ótico, (b) trilha ótica

- **Controle de Direção por Sensores Óticos ("AAT Sistemas de Movimentação") :** Nesta aplicação o *AGV* possui um conjunto de sensores óticos, que adaptados na parte inferior do veículo, identificam a presença da fita-guia enviando sinais ao *CLP* que efetua automaticamente a correção na direção do *AGV*, colocando-o novamente sobre a trilha. Este sistema possui três sensores (Fig. 5.3), sendo: (1) Um

para alinhamento à direita, (2) outro para alinhamento à esquerda (3) e um terceiro (central) para presença de fita. Quando é utilizada uma fita metálica pode ser usado um quarto sensor indutivo para segurança que opera em conjunto com o sensor de presença ótica.

Figura 5.3 - Sensor ótico.

5.2.2 - Sistemas de Caminhos Dinâmicos

Estes sistemas são mais flexíveis e suportam facilmente modificações de layout e requisitos variantes de capacidade de planta, mas estes sistemas são mais complexos e custosos. Navegação por visão computacional por laser ou por *GPS,* só em aplicações outdoor, são exemplos de sistemas de caminhos dinâmicos.

- **Sistema Laser :** Dentro dos sistemas de navegação, os de navegação por laser (Figura 5.4) é o que tem tido mais implantação no mercado, com tendência a substituir rapidamente o sistema filoguiado. Neste tipo de sistema o *AGV* é equipado como um laser scanner, que realiza constantemente uma varredura rotativa, detectando a posição de painéis refletores colocados ao longo do *layout*. A posição do *AGV* é determinada com base na triangulação dos feixes de luz refletidos. Este sistema permite re-configurar facilmente o *layout*, bastando para isso alterar a colocação dos painéis refletores e re-programar a configuração do *AGV*.

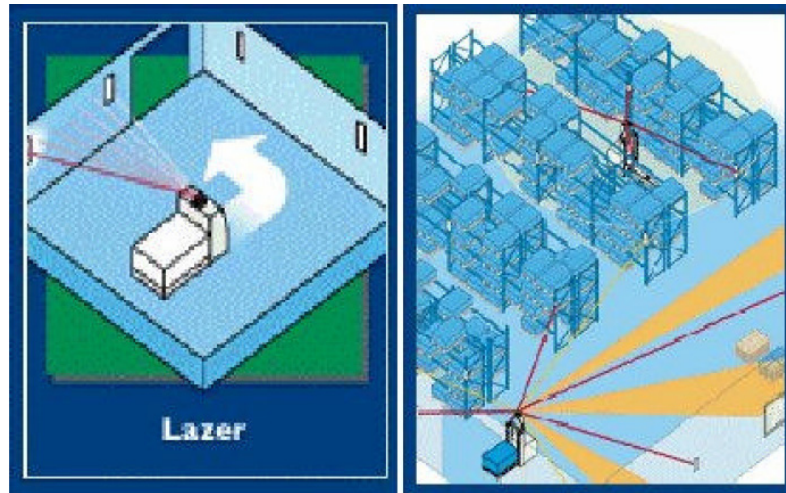



Figura 5.4 - Sistema Laser (Rocha, 2001)

- **Sistema GPS**: É utilizado principalmente em áreas externas, em veículos de grande porte, como caminhões, carros, gruas, etc. Este sistema permite um bom controle apenas em percursos grandes, pois a sua precisão é baixa (da ordem dos metros), o que não permite a carga ou descarga automática a partir de outros métodos. O sistema tem que ser complementado a partir de outros sistemas, como por exemplo o de detecção de obstáculos. A Figura 5.5 mostra uma idéia do funcionamento deste sistema.

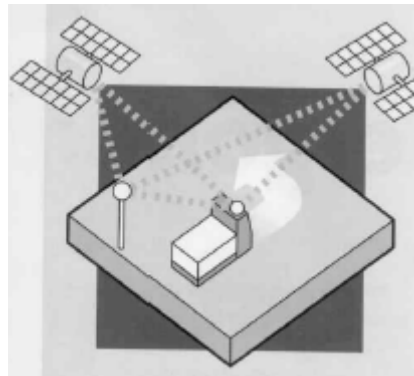



Figura 5.5 – Localização em *AGV's* – GPS (Leitão, P., 2005)

Além dos sistemas de localização mencionados existem outra grande variedade de sistemas de localização de *AGV's*, quais sejam: sistemas de comunicação por radio freqüência, sistemas de localização por coordenadas, ultra-som, etc.

6 - NAVEGAÇÃO ROBÓTICA

6.1 – O PROBLEMA DA NAVEGAÇÃO

A navegação é um processo usado por um robô móvel para movimentar-se desde uma posição inicial a uma posição final com base a um ponto de referencia (Ribeiro, et al., 2002).

Leonard e Durrant – White (1991) resumiram o problema geral da navegação dos robôs moveis através de três questões: onde eu estou ?, onde eu estou indo? e como eu conseguiria chegar lá ? (Borenstein, et al., 1995)

A navegação envolve três tarefas básicas: mapeamento, planejamento e direção. Um processo de alto nível, denominado de planejamento de tarefas, especifica o destino e restrições para o sistema, como o tempo (Tourino, 2002).

A primeira parte da navegação, mapeamento, é realizada através do uso de mapas pré-armazenados ou de mapas "aprendidos" pelo sistema sensorial à medida que o robô atravessa o ambiente. A modelagem do ambiente é realizada pela análise dos dados sensoriais para a construção e modificação dos mapas.

A segunda tarefa é o planejamento feito através da procura de caminhos possíveis no mapa construído. Caso não exista um mapa, o caminho é encontrado entre os objetos sentidos pelo sistema de sensoriamento no momento da construção do mapa. A melhor alternativa é então escolhida de forma a atender às restrições impostas pela tarefa.

A terceira tarefa realizada na navegação é a guiagem do robô através do caminho definido anteriormente. O movimento do robô é então controlado utilizando-se seu modelo cinemático e dinâmico. Durante o movimento do robô, o processo de percepção examina continuamente os dados sensoriais a fim de detectar colisões potenciais. Quando uma colisão é possível o processo de percepção inicia uma ação evasiva. O nível final de segurança em um robô móvel é obtido através da detecção de colisão por sensores de toque ou contato. A Figura 6.1 mostra a inter-relação hierárquica entre as funções de navegação de um robô móvel

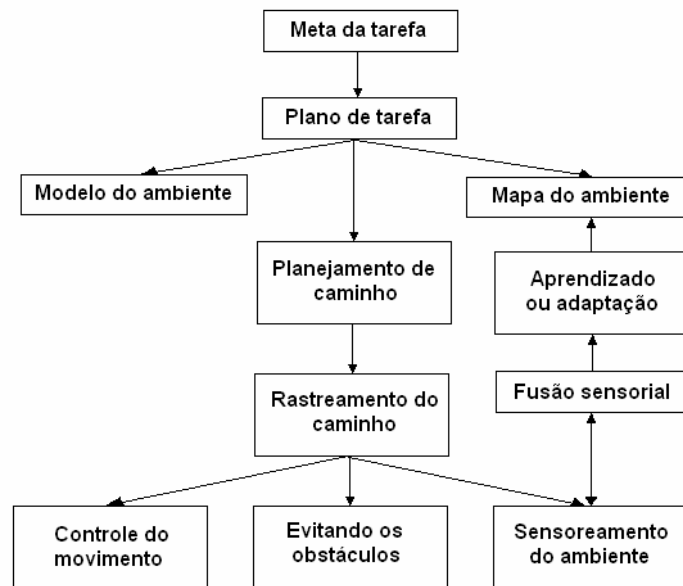



Figura 6.1 - Hierarquia das funções de navegação de um robô móvel (modificado - Ribeiro et al., 2002)

6.2 – TÉCNICAS DE NAVEGAÇÃO UTILIZADAS EM ROBÓTICA

As técnicas utilizadas em navegação robótica podem ser classificadas em dois grandes grupos: (1) medidas relativas da posição e (2) medidas absolutas da posição (Borenstein, et al., 1995).

6.2.1 – Navegação baseada em medidas relativas da posição

- **Navegação Odométrica**: Este método usa encoders para medir a rotação das rodas e controlar a orientação do veículo. A vantagem deste sistema de navegação é sua autonomia e a capacidade de prover sempre ao veículo uma estimativa de sua posição. A desvantagem deste sistema é que o erro de posição aumenta sem limites, a menos que uma referência independente seja usada periodicamente para reduzir o erro.

- **Navegação Inercial**: Este método usa giroscópios e algumas vezes acelerômetros para tomar medidas das rotações e acelerações do veículo. As medidas são integradas para obter uma estimativa da velocidade e posição do robô. Este sistema tam-

bém tem a vantagem de ser autônomo. Já uma desvantagem nestes sistemas é o alto custo dos sensores. Os giroscópios, com tecnologia recente, usam fibra ótica, conhecidos também como giroscópios a laser e são de muita precisão.

6.2.2 – Navegação baseada em medidas absolutas da posição

- **Navegação por sinais ativos** ("*active beacons*"): Este método computa a posição absoluta do robô através da medida da direção de incidência de três ou mais sinais ativos transmitidos. Os transmissores usualmente usam luzes ou freqüências de radio e são localizados em lugares conhecidos do ambiente.

- **Navegação por reconhecimento de marcas artificiais**: Este método usa marcas artificiais distintivas as quais são localizadas em lugares conhecidos do ambiente de navegação. A vantagem destas marcas artificiais é que podem ser projetadas para serem otimamente detectadas e reconhecidas, mesmo em ambientes com condições adversas.

- **Navegação por reconhecimento de marcas naturais**: Este método utiliza marcas com características próprias distintivas do ambiente, o qual não precisa ser preparado. Este implica que o ambiente tem que ser muito bem conhecido. A confiabilidade deste método não é tão alta como o método onde se utilizam marcas artificiais.

- **Navegação por comparação de modelos** ("*Model Matching*"): Através deste método a informação adquirida pelos sensores do robô é comparada com um mapa modelo do ambiente. Se as características do mapa criado com base nos sensores do robô e o mapa modelo do ambiente coincidem, então a posição absoluta do robô pode ser estimada. Os mapas usados em navegação podem ser classificados em dois tipos: (1) mapas geométricos e (2) mapas topológicos. Mapas geométricos representam o espaço de navegação total, denominado na literatura inglesa como "*world (W)*", em um sistema de coordenadas globais, em quanto os topológicos representam o espaço de navegação em uma rede de nós e arcos.

6.3 – PLANEJAMENTO DO MOVIMENTO

O objetivo principal no planejamento de movimento é avaliar os caminhos livres de colisões desde a posição inicial até a posição final, na navegação do robô móvel, tomando-se em conta os aspectos geométricos, físicos, e de tempo envolvidos neste processo.
Há três diferentes aspectos a serem analisados no planejamento do movimento (Ribeiro et al., 2002):

- Planejamento da trajetória ("*Path Planning*")
- Planejamento de manobras de movimento ("*Manoeuvre Planning*")
- Geração da trajetória ("*Trajectory Generation*")

Um caminho ("*Path*") é o lugar geométrico dos pontos num espaço dado onde o robô tem que passar. A trajetória refere-se a um caminho no qual uma lei temporal é especificada (aceleração e velocidade em cada ponto).

6.3.1 Planejamento da Trajetória

Segundo Ribeiro (2002), as metodologias desenvolvidas para o planejamento de trajetória podem ser classificadas em dois grupos: (1) Métodos globais e (2) Métodos Locais. Os métodos globais estão formados pelos seguintes métodos: mapa de caminhos (*roadmap*), decomposição celular (*cell Decomposition*). Estes métodos dão a habilidade ao robô para se localizar em termos absolutos (*GPS – Global Positioning System*) ou em relação a um sistema de mapas, e mover-se para um ponto desejado. O método dos campos potenciais é classificado dentro dos métodos locais. Estes métodos dão a capacidade ao robô de determinar sua posição relativa a objetos próximos (estacionários ou não) e interagir com estes de forma correta.

Na seqüência serão apresentados brevemente cada um dos método tanto globais como locais acima tratados.

- **Mapa de caminhos ("*Roadmap*")**: Partindo de um espaço livre, através deste método, (C_{free}) um gráfico de caminhos pode ser definido (*roadmap*). As diferentes

formas de obter um "*roadmap*" são através de gráficos de visibilidade, diagramas de Varonoi, caminhos livres (*freeway*) e silhueta (*silhouette*).

- **Decomposição Celular**: Este método é baseado na decomposição do espaço livre (C_{free}) do robô em simples regiões ou células.
- **Campos Potenciais:** O robô é tratado como uma partícula atuando sobre a influência de um campo potencial "U". A direção ao objetivo é modelada por um campo atrativo. Os obstáculos são evitados por ação de uma força repulsiva gerada por um campo negativo.

7 – VISÃO COMPUTACIONAL

O sistema de navegação do robô móvel, utilizado como um *AGV* na *FMC*, é baseado em um sistema de visão. Este sistema é composto de uma serie de algoritmos executados seqüencialmente para cumprir tarefas pré-determinadas de navegação dentro da *FMC*. Neste capitulo são apresentados conceitos básicos de processamento de imagens e visão computacional utilizados no desenvolvimento do trabalho.

7.1 - CONCEPÇÃO DE UM SISTEMA DE VISÃO ARTIFICIAL

Os sistemas de visão computacional visam, com ajuda do conhecimento de diversas áreas (mecânica, biologia, medicina, comunicação visual, eletrônica, matemática, etc.), obter um conjunto de técnicas e metodologias que possam dar suporte ao desenvolvimento de teorias e produtos suficientemente eficientes e confiáveis para aplicações práticas. Cita-se, como exemplo, a automatização dos processos de controle de qualidade, identificação e classificação de produtos e exploração de ambientes diversos (Facon, 2002).

O objetivo da visão computacional é recuperar informações úteis de uma cena a partir de projeções bidimensionais contidas em imagens (Jain, et al., 1995).

O processamento e a análise de imagens é uma ciência que permite modificar, analisar e manipular imagens digitais, originalmente continuas, a partir de um computador.

Um sistema de visão é constituído de:

- **Sensores de Visão**: fornecem uma projeção da cena de trabalho e realizam a aquisição de imagens. A câmera é o sensor geralmente utilizado nos sistemas de visão. Para o bom desempenho deste sensor e recomendável trabalhar com uma boa iluminação que permite a obtenção de uma imagem de melhor contraste e de melhor qualidade, reduzindo portanto a quantidade de processamentos preliminares.
- **Hardware de Digitalização de Imagens:** O objetivo deste módulo é colocar a imagem do sensor na memória. O módulo de aquisição permite escrever de modo eficiente na memória que pode ser lida pelo computador e pelo módulo de visuali-

zação. O módulo de digitalização transforma as imagens contínuas em imagens digitais.

- **Computador:** Esta parte do sistema de visão executa os algoritmos de processamento de imagens permitindo flexibilidade e custos de processamento e de memória relativamente baixos.

7.2 – APLICAÇÕES

Aplicações de processamento e análise de imagens podem ser classificadas a partir de critérios como a possibilidade de um conhecimento inicial prévio do contexto ou não, a necessidade de processamento em tempo real, etc. A seguir são apresentados exemplos de aplicações que envolvem técnicas de processamento de imagens.

7.2.1 – Aplicações Industriais

De forma geral, o desenvolvimento de um projeto industrial que exija a utilização de técnicas de processamentos de imagens supõe um importante conhecimento inicial do produto (normas de concepção, regras de produção, etc.) e exige técnicas rápidas para atender processos em tempo real.

- **Controle de um robô**: controlar um robô é uma das tarefas que requer rapidez e precisão. Um robô é usado para manipular peças que podem ser ordenadas ou montadas com outros tipos de peças. Estas podem já estar acondicionadas em locais previamente conhecidos ou não. O alvo do robô, atingido com auxilio de visão computacional, pode envolver escolha de uma peça entre varias outras peças ou a determinação de uma trajetória para evitar obstáculos.
- **Inspeção visual**: O controle de qualidade de um produto é uma tarefa essencial no domínio industrial. O processo de inspeção implica na medição de determinadas propriedades de um produto, como dimensões geométricas, superfícies, posição, orientação, etc. A inspeção automatizada, mediante técnicas de processamentos de imagens possibilita a quantificação de propriedades e coleta de dados sobre o produto que um inspetor humano não é capaz de realizar.

7.2.2 – Reconhecimento de padrões

A exigência crescente na automatização de tarefas humanas repetitivas e / ou cansativas, levou os pesquisadores a desenvolver ferramentas especificas para atender projetos cujo conhecimento inicial é pobre e / ou incompleto e onde um certo grau de inteligência é importante. O reconhecimento de um padrão realiza este tipo de tarefas. Entre algumas das aplicações podemos citar: o reconhecimento de caracteres, o reconhecimento de impressões digitais, reconhecimento de assinaturas, etc.

7.2.3 – Reconstrução tridimensional

A constante evolução de desempenho dos computadores viabiliza a percepção tridimensional do mundo a partir de imagens bidimensionais. Como exemplo podemos citar a tomografia de ressonância magnética, com imagens de alta resolução e nitidez a qual permite esperar uma explosão de ferramentas de visualizações tridimensionais de diferentes partes do corpo humano e o desenvolvimento de sistemas de apoio ao diagnóstico médico (Facon, 2002).

7.3 – ARQUITETURA DE UM SISTEMA DE VISÃO COMPUTACIONAL

Os processamentos das imagens, capturadas por um sistema de visão, mudam muito em função da área de trabalho. Assim, estes processamentos variam sobre as imagens segundo as seguintes característica:

- **A natureza das imagens**: depende do meio onde elas são capturadas, podendo ser áreas como medicina, do meio industrial, ou de laboratório, onde as entidades encontradas e informações contidas são muito diversas.

- **A qualidade das imagens**: A qualidade requerida de uma imagem varia de acordo com a aplicação. A resolução da imagem depende principalmente das dimensiones da matriz, do número de níveis de cinza de cada pixel e do intervalo entre imagens. As condições de iluminação do ambiente, é outra característica importante que influenciam na qualidade da imagem.

- **Conhecimento do ambiente**: O conhecimento inicial do ambiente é elemento importante no processo de análise. No meio industrial o conhecimento inicial, sempre existente, permite uma análise mais dirigida para uma solução rápida. Já em casos onde o conhecimento inicial é fraco, como no caso de imagens adquiridas por satélite, o processo de analise deve ser mais completo para suprir a falta de conhecimento.

O processamento digital de imagens pode ser classificado em três níveis distintos: baixo, médio e alto. No processamento de baixo nível os dados de entrada são pixéis da imagem original capturada e os dados de saída representam propriedades da imagem, na forma de valores numéricos associados a cada pixel. No processamento de nível médio este conjunto de valores produz como resultado uma lista de características. O processamento de alto nível, produz a partir destas características, uma interpretação do conteúdo da imagem (Rosenfeld, 2001). Uma estrutura funcional completa de um sistema de processamento e analise de imagens, classificado em níveis, é apresentado na Figura 7.1.

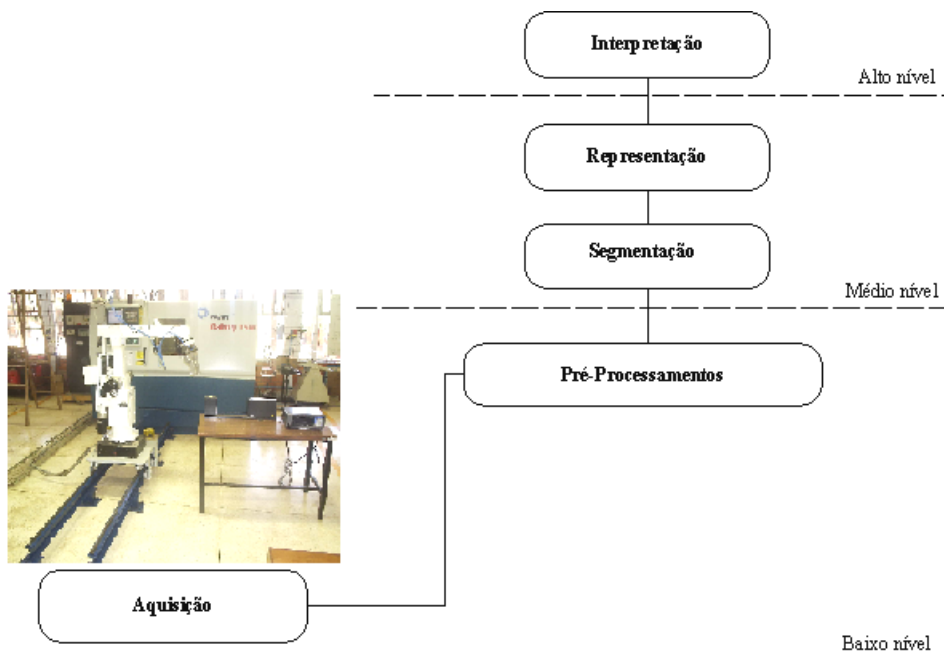



Figura 7.1 - Estrutura funcional de um sistema de visão computacional (modificado - Gonzalez, 2000)

No baixo nível podem ser analisadas duas etapas: (1) Aquisição e digitalização e (2) Pré-processamentos.

- **Aquisição e digitalização**: A imagem do sensor é transformada em uma imagem digital sobre a forma de uma matriz de valores inteiros chamados pixels. O sensor poderia ser uma câmera de vídeo monocromática ou colorida analógica e uma placa de digitalização (*frame grabber*). No caso de câmeras de vídeo digitais há necessidade somente de uma interface com o computador. A natureza do sensor e da imagem que ele produz são determinadas pela aplicação.

- **Pré-processamento**: O objetivo nesta etapa e corrigir os defeitos e imperfeições aparecidos durante a aquisição da imagem, que podem ter como causa características físicas do sistema, condições deficientes de iluminação, etc.

As duas etapas envolvidas no nível médio são as seguintes: (1) Segmentação e (2) Representação.

- **Segmentação**: O objetivo desta etapa é dividir a imagem em partes constitutivas, chamadas de regiões. Em uma imagem natural, a segmentação é efetuada pela detecção de descontinuidades (contornos) e / ou de similaridades (regiões) na imagem.

- **Representação**: O objetivo desta etapa é elaborar uma estrutura adequada, agrupando resultados das etapas precedentes a o armazenamento dos diversos padrões que contem o conhecimento a priori. Ela permite também medir as propriedades das "formas" resultantes da segmentação.

A última etapa que pertence ao alto nível é da interpretação. Esta etapa fecha o ciclo dos processos que são executadas no processamento de uma imagem segundo a arquitetura apresentada num sistema de visão computacional.

- **Interpretação**: Esta etapa é a parte inteligente do processo

Outra forma de representar as etapas principais que conformam a arquitetura de um sistema de visão computacional é mostrada na Figura 7.2 (Fu, *et al*.,1987) e (Gonzalez, et al.,

2000). Estas etapas são: Sensoriamento (Aquisição), Processamento de Imagens (Pré-processamento), Segmentação, Descrição, Reconhecimento e Interpretação.

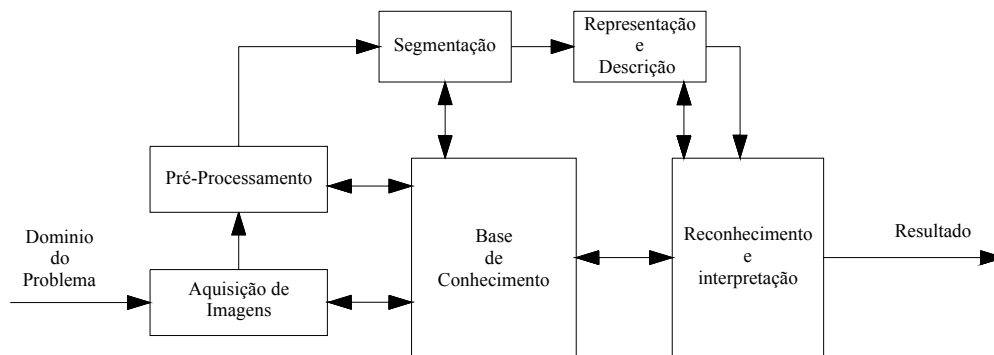



Figura 7.2 - Etapas básicas de um sistema de visão computacional
(modificado - Fu, *et al.*,1987)

7.4 – PROCESSAMENTO DE IMAGENS

O processamento de imagens é uma área que abarca muitos aspectos e técnicas, dependendo das aplicações e / ou do campo de estudo onde são utilizados os sistemas de visão para a aquisição das imagens. A maioria dos algoritmos de visão exigem algum processamento preliminar de imagem. O objetivo do processamento digital de imagens é realizar transformações de forma que se consiga uma melhora do aspecto visual de certas feições estruturais para a análise posterior e interpretação. Num processamento digital de imagens incluem realce (melhora da qualidade da imagem adquirida), compressão (compactação de imagens digitais para transmissão), restauração (eliminação dos efeitos de degradação), e extração de características (localização de elementos especiais da imagem, como contornos, e regiões) (Trucco, 1998).

Segundo K. Huh (2005), o processo de aquisição de imagens pode ser considerado como uma transformação desde o espaço real 3D conhecido como “mundo” a um plano 2D de uma imagem.

As principais áreas de aplicações do processamento digital de imagens que requerem métodos capazes de realçar as informações contidas nas imagens para a posterior interpreta-

ção e análise são: análise de recursos naturais (geologia, agricultura, cartografia, etc.), análise ambiental (monitoramento de poluição e aquecimento global, planejamento urbano), meteorologia (movimento de nuvens, previsão de tornados), biomedicina, aplicações industriais, navegação robótica, etc.

7.4.1 – IMAGEM

O termo imagem refere-se a uma função de intensidade luminosa bidimensional, denotada por $f(x,y)$, em que o valor ou amplitude de f nas coordenadas espaciais (x,y) dá a intensidade $(brilho)$ da imagem naquele ponto (Gonzalez, et al., 2000). Como a luz é uma forma de energia, $f(x,y)$ deve ser positiva e finita, isto é:

$$0 < f(x,y) < \infty \tag{7.1}$$

No caso de uma imagem que possui informações em intervalos ou bandas distintas de freqüência, é necessário uma função $f(x,y)$ para cada banda. É o caso de imagens coloridas padrão *RGB*, que são formadas pela informação de cores primarias, como o vermelho "*Red*", verde "*Green*" e azul "*Blue*". Para o processamento da imagem digitalizada, é fundamental representar sua informação num formato adequado ao tratamento computacional.

Uma imagem pode ser representada por uma matriz em que os índices de linha e coluna referenciam o brilho médio amostrado no ponto correspondente na cena.

As imagens que as pessoas percebem em atividades visuais consistem em luz refletida dos objetos. A natureza básica de $f(x,y)$ pode ser caracterizada por dois componentes: (1) a quantidade de luz incidindo na cena sendo observada (*iluminação*) e (2) a quantidade de luz refletida pelos objetos da cena (refletância) e são representados por $i(x,y)$ e $r(x,y)$ respectivamente. O produto das funções $i(x,y)$ e $r(x,y)$ resulta $f(x,y)$ (Gonzalez, et al., 2000) :

$$f(x,y) = i(x,y)r(x,y) \tag{7.2}$$

onde

$$0 < i(x,y) < \infty \tag{7.3}$$

$$0 < r(x, y) < 1 \qquad (7.4)$$

A equação (7.4) indica que a refletância é limitada entre 0 (absorção total) e 1 (refletância total). A natureza de $i(x, y)$ é determinada pela fonte de luz, e $r(x, y)$ é determinada pelas características dos objetos na cena.

7.4.2 - TÉCNICAS BÁSICAS PARA A MODIFICAÇÃO OU REALCE DE IMAGENS

7.4.2.1 - Histograma

O realce e análise das imagens são feitos através de informações fornecidas pelo histograma da imagem adquirida. O histograma de uma imagem informa aproximadamente a distribuição de probabilidade de cada um dos níveis de cinza de uma imagem. Já segundo Gonzalez (2000) o histograma de uma imagem digital com níveis de cinza no intervalo $[0, L-1]$ é uma função discreta $p(r_k) = n_k / n$ em que r_k é o k-ésimo nível de cinza, n_k é o número de pixéis na imagem com esse nível de cinza, n é o número total de pixels na imagem e $k = 0,1,2,..., L-1$. A Figura 7.3 mostra a imagem original com seu respectivo histograma. Notar na Figura 7.3 (b) os três picos com maior numero de pixéis para determinados níveis de cinza da imagem nas faixas 50-100 e 200-250.

a) b)

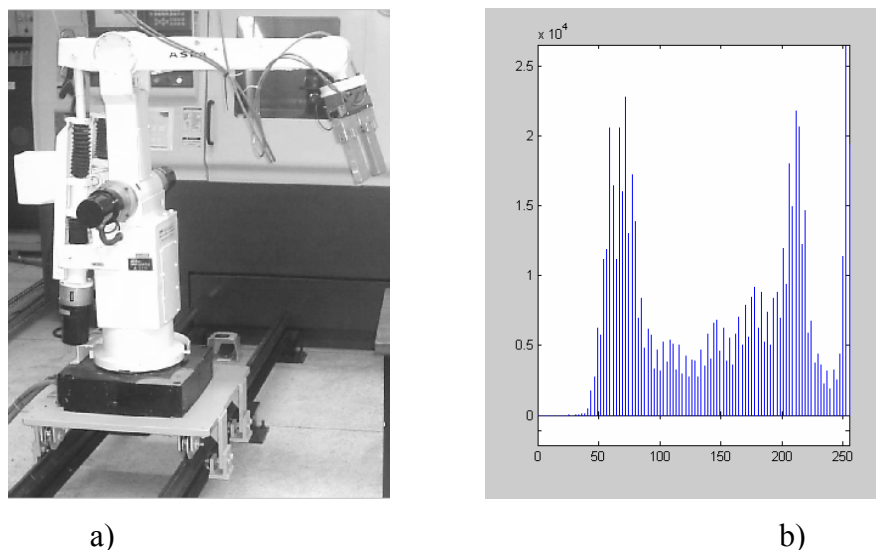



Figura 7.3 – Histograma de uma imagem. (a) Imagem original. (b) Histograma correspondente

7.4.2.2 - Equalização de Histogramas

A equalização de histograma consiste basicamente em uma redistribuição dos valores de níveis de cinza para os píxeis, resultando assim numa imagem com maior contraste, e como conseqüência, de melhor visualização. A equalização modifica o histograma da imagem original de tal forma que a imagem transformada tenha um histograma uniforme. Os níveis de cinza devem aparecer na imagem com a mesma freqüência (Ramesh, et al., 1995).

No caso de uma imagem original as freqüências são dadas por:

$$p_r(r_i) = n_i / n \qquad (7.5)$$

onde i representa o nível de cinza e $i \in [0, L-1]$, n_i corresponde ao numero de pixels do nível de cinza i e n corresponde ao número total dos pixels.

A transformação associada à uniformização é:

$$s_k = T(r_k) = \sum_{j=0}^{k} \frac{n_j}{n} = \sum_{j=0}^{k} p_k(r_k) \qquad (7.6)$$

A transformação inversa é:

$$r_k = T^{-1}(s_k) \qquad (7.7)$$

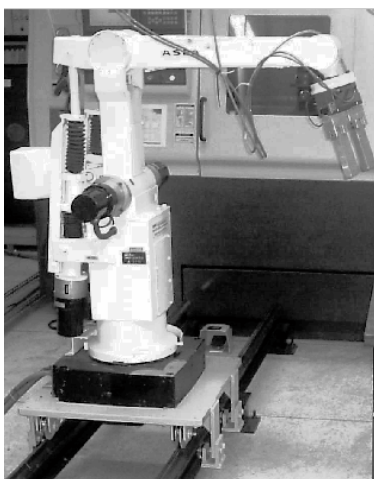
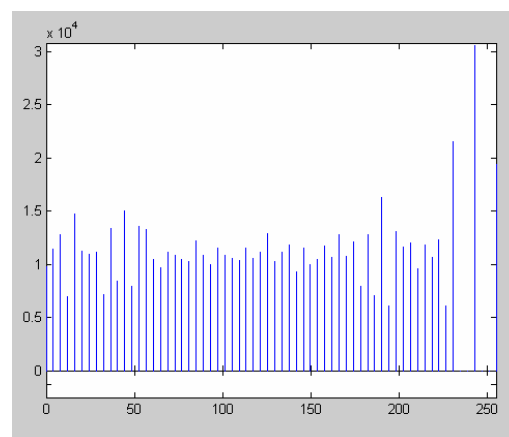
Ao processo geral denomina-se de equalização do histograma. Por causa dos problemas de aproximação de densidade, o histograma obtido não corresponde exatamente ao histograma de uma lei uniforme. A Figura 6.4 mostra a imagem equalizada com seu respectivo histograma.

a) b)

Figura 7.4 – Equalização do histograma de uma imagem. (a) Imagem equalizada. (b) Histograma correspondente

Outro algoritmo de equalização do histograma, proposto por (Haralick, 1988), fixa o número k de níveis de cinza em um número menor ao valor de origem, permitindo aproximar-se mais da lei uniforme.

7.4.2.3 - Filtragem no Domínio Espacial

O uso de máscaras espaciais para processamento de imagens é usualmente chamado de filtragem espacial (Gonzalez, 2000). Nesta parte do trabalho serão considerados filtros espaciais lineares e não-lineares para o realce das imagens.

Os filtros espaciais podem ser classificados em passa-baixa, passa-alta ou passa-banda. Os dois primeiros são os mais utilizados em processamento de imagens. O filtro passa-banda é mais utilizado em processamentos específicos, principalmente para remover ruídos periódicos. Os filtros espaciais podem ser divididos em filtros lineares e filtros não-lineares.

Filtros Lineares

- **Filtro Passa – Baixa**

  Os denominados filtros passa-baixa atenuam ou eliminam os componentes de alta freqüência no domínio de Fourier enquanto deixam as freqüências baixas inalteradas. Os componentes de alta freqüência caracterizam bordas e outros detalhes finos de uma imagem de forma que o efeito resultante da filtragem passa-baixa é o borramento da imagem. Portanto o efeito visual desse tipo de filtro é o de "suavização" (*smoothing*) da imagem uma vez que as altas freqüências que correspondem as transições abruptas, são atenuadas. A suavização tende também, pelas mesmas razões, a minimizar o efeito do ruído em imagens.

  Segundo Gonzáles (2000), a forma de resposta impulsiva necessária para implementar um filtro espacial passa-baixa (suavização) indica que o filtro tem que possuir todos os coeficientes positivos. Para um filtro espacial de 3 x 3 a estrutura mais simples seria uma máscara na qual todos os coeficientes tenham valor 1. Uma solução de máscara de dimensão 3 x 3 normalizada é mostrada abaixo. Mascaras maiores normalizadas são também mostradas na Figura 7.5.

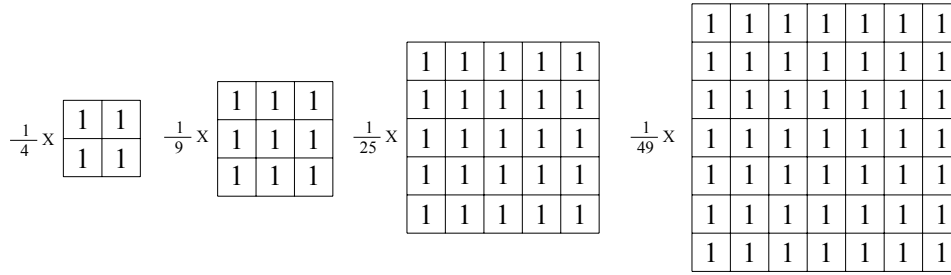



Figura 7.5 – Filtros passa-baixas espaciais de vários tamanhos

O valor atualizado do pixel $P_A(x,y)$ após a passagem da máscara qualquer M, de dimensão $(2m+1)x(2m+1)$, é realizado através da utilização da equação (7.8), conhecida também como convolução discreta (Trucco, 1998). A Figura 6.6 mostra a filtragem da uma imagem, afetada por ruído a um 20%, com o filtro passa baixa.

$$P_A(x,y) = \sum_{i=-m}^{m} \sum_{j=-m}^{m} P(x+i, y+j).M(i,j) \tag{7.8}$$

onde:

$P_A(x,y)$	Valor atualizado do pixel
$P(x,y)$	Valor original do pixel em (x,y)
$M(i,j)$	Valor da máscara na posição (i,j)
i, j	Variáveis do somatório

a) b) c)

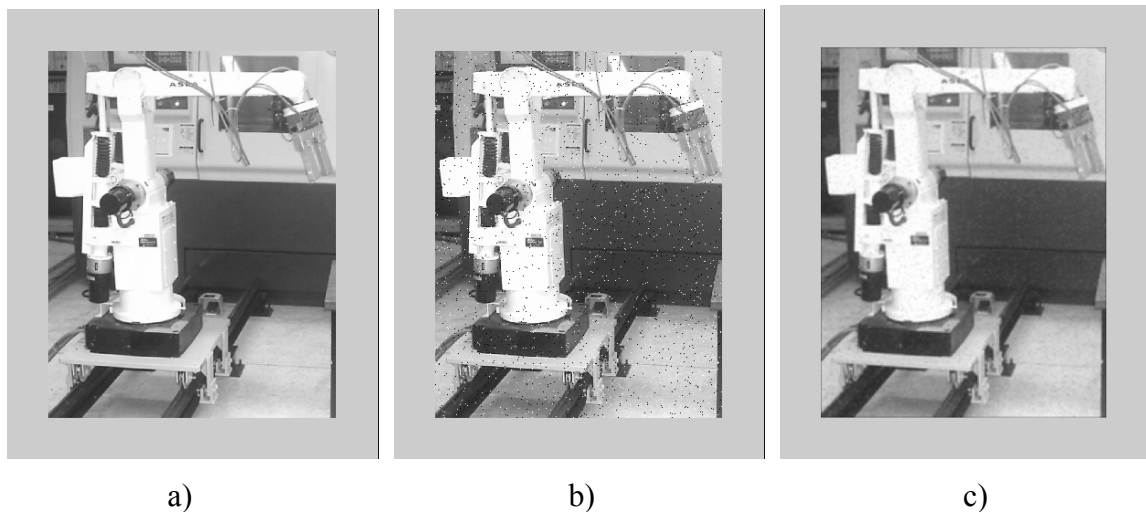



Figura 7.6 – Imagem filtrada com filtro passa baixa. (a) Imagem original , (b) adição de ruído *salt&pepper*, (c) Imagem filtrada - filtro 5 x 5 – "*smoothing*".

- **Filtro Passa – Alta (Básica)**

A filtragem passa-alta atenua ou elimina os componentes de baixa freqüência. O efeito visual deste tipo de filtro é o de agudização (*sharpening*) da imagem. As transições entre diferentes regiões, bordas e detalhes finos da imagem tornam-se mais nítidas. Como conseqüência o ruído, que possa existir numa imagem, também pode ser enfatizado.

A forma da resposta ao impulso necessária para implementar um filtro espacial passa-alta (aguçamento) indica que o filtro deve ter coeficientes positivos próximos ao seu centro, e coeficientes negativos na periferia. Para uma máscara 3 x 3, como a que é mostrada na Figura 7.7, a escolha de um valor positivo no centro, com coeficientes negativos no resto da máscara, satisfaz essa condição.

$$\frac{1}{9} \times \begin{array}{|c|c|c|} \hline -1 & -1 & -1 \\ \hline -1 & 8 & -1 \\ \hline -1 & -1 & -1 \\ \hline \end{array}$$

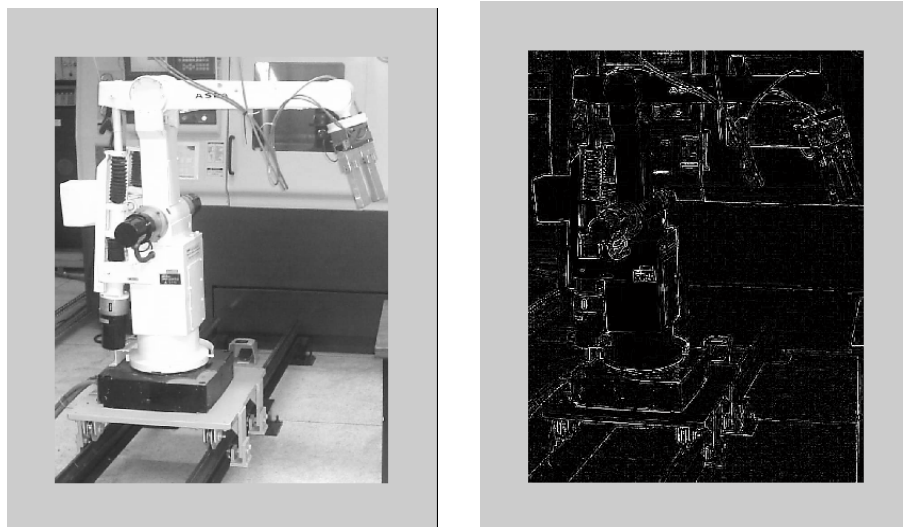



Figura 7.7 – Imagem filtrada com um filtro passa-alta

Filtros Não – Lineares

Os filtros não lineares, conhecidos também como filtros por derivadas, são utilizados para alterar a imagem sem diminuir sua resolução.

Conforme Gonzáles (2000) o cálculo da média dos pixéis sobre uma região tende a borrar os detalhes de uma imagem. Assim como o cálculo da média é análogo à integração, pode-se esperar que a diferenciação tenha o efeito oposto de agudização da imagem.

O gradiente é o método mais comum de diferenciação em aplicações de processamento de imagens. Para uma função $f(x, y)$ o gradiente de f nas coordenadas (x, y) é definido como o vetor:

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \quad (7.9)$$

a equação (7.10) mostra a magnitude desse vetor

$$\nabla f = mag(\nabla f) = \left[\left(\frac{\partial f}{\partial x} \right)^2 + \left(\frac{\partial f}{\partial y} \right)^2 \right]^{1/2} \quad (7.10)$$

Esta equação é a base para várias abordagens de diferenciação de imagens. Considere a região da imagem mostrada na Figura 7.8 (a), em que os $z's$ denotam os valores dos níveis de cinza. A equação (7.10) pode ser aproximada no ponto z_5 de várias maneiras. A mais simples consiste em usar a diferença $(z_5 - z_8)$ na direção x e $(z_5 - z_6)$ na direção y combinadas como:

$$\nabla f = \left[(z_5 - z_8)^2 + (z_5 - z_6)^2 \right]^{1/2} \quad (7.11)$$

A equação (7.11) pode ser substituída pela equação (7.12) em valores absolutos e obter resultados similares.

$$\nabla f = |z_5 - z_8| + |z_5 - z_6| \quad (7.12)$$

Uma outra abordagem, segundo Gonzáles (2000) para aproximação da equação (7.10) é usar as diferenças cruzadas como é mostrado na equação (7.13).

$$\nabla f = \left[(z_5 - z_9)^2 + (z_6 - z_8)^2\right]^{1/2} \quad (7.13)$$

ou, usar os valores absolutos,

$$\nabla f = |z_5 - z_9| + |z_6 - z_8| \quad (7.14)$$

As equações (7.11) e (7.12) podem ser implementadas através do uso de máscaras de tamanho $2x2$. A equação (7.14) pode ser implementada tomando-se o valor absoluto das respostas das duas máscaras mostradas na Figura 7.8 (b) chamadas de operadores cruzados de gradiente de Roberts.

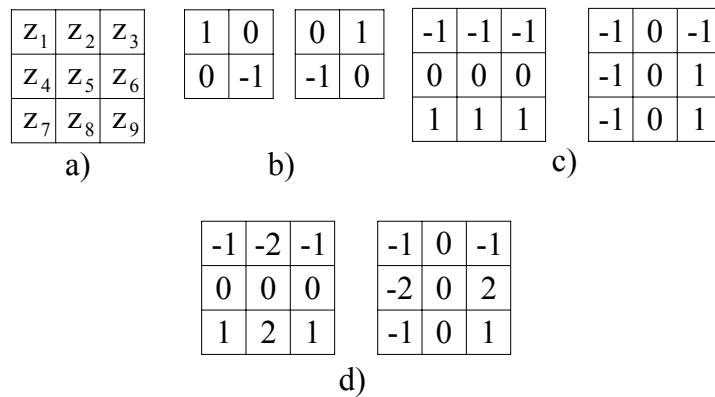



Figura 7.8 – Máscaras de filtros não lineares. (a) Uma região 3 x 3 de uma imagem (os z's são valores de nível de cinza), (b) Máscaras de Roberts, (c) Máscaras de Prewitt (d) Máscaras de Sobel (modificado – Gonzalez, 2000).

Máscaras de tamanhos pares são incômodas de se implementar. Uma aproximação para a equação (7.10) ainda no ponto z_5 mas agora usando uma vizinhança de 3 x 3, é

$$\nabla f = |(z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)| + |(z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)| \quad (7.15)$$

As mascaras mostradas na Figura 7.8 (c) chamadas de operadores de Prewitt, podem ser usadas para implementar a equação (7.15). Finalmente a Figura 7.8 (d) mostra ainda outro par de máscaras, chamados operadores de Sobel, para aproximação da magnitude do gradiente.

- **Operador de Roberts**

O operador gradiente de Roberts (Gonzalez, 2000) é o método mais simples dos filtros não lineares utilizado para detecção de contornos ou bordas. Este método apresenta uma desvantagem, dependendo da direção, certos contornos podem ser mais realçados que outros. Como resultado de sua aplicação, obtém-se uma imagem com altos valores de nível de cinza, em regiões de limites bem definidos e valores baixos em regiões de limites suaves, sendo "0"para regiões de nível de cinza constante.

O operador, formado por duas máscaras, é mostrado na Figura (b), onde G_x é a máscara que detecta as variações do gradiente na direção horizontal e G_y é a máscara que detecta as variações do gradiente na direção vertical. A Figura 7.9 mostra uma aplicação simples do operador de Roberts.

Figura 7.9 – Imagem filtrada utilizando o operador de Roberts

- **Operador de Sobel**

O operador de Sobel tem a propriedade de realçar linhas verticais e horizontais mais escuras que o fundo, sem realçar pontos isolados(Gonzalez et al., 2000). Este operador está formado por duas máscaras, as quais são mostradas na Figura 7.8 (d). A Figura 7.10 ilustra o resultado obtido através da aplicação do operador de Sobel

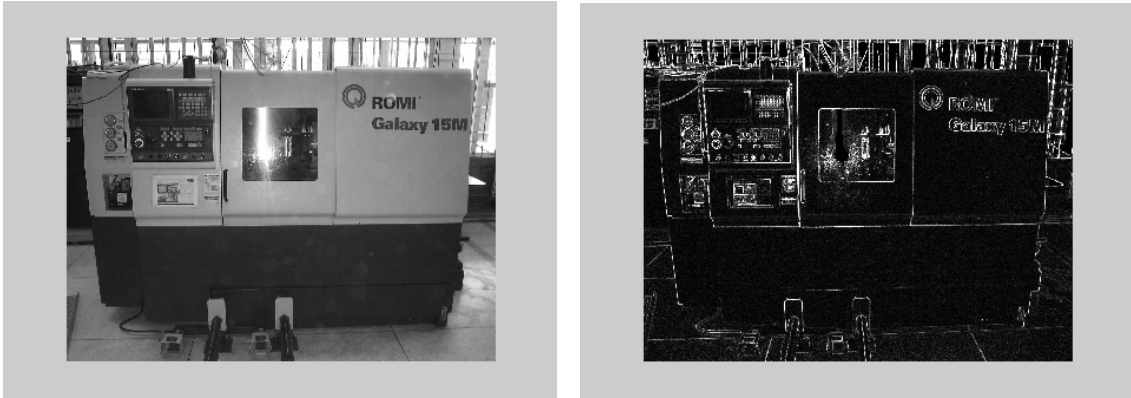



Figura 7.10 – Imagem filtrada utilizando o operador de Sobel

- **Operador Laplaciano**

A função que representa o Laplaciano, aplicada a uma imagem digital, é mostrada na equação (7.16). O efeito da aplicação desta função numa imagem evidencia uma informação importante que é o ponto onde a função passa pelo zero (*zero-crossing*), dado que tais pontos são localizações de bordas.

$$\Delta^2 I(x,y) = L[I(x,y)] = \frac{\partial^2 I(x,y)}{\partial x^2} + \frac{\partial^2 I(x,y)}{\partial y^2} \qquad (7.16)$$

A intensificação das bordas pode ser realizada por subtração da imagem $I(x,y)$ com a derivada segunda de $I(x,y)$. Uma operação bem conhecida no campo da fotografia, onde é aplicada esta operação, é chamada "*unsharp masking*" $(I - \Delta^2 I)$.

Para uma imagem digital, a função do Laplaciano (Eq. 7.17) calculada num ponto pode ser aproximado da seguinte maneira:

$$\begin{aligned}\Delta^2 I(x,y) &= L[I(x,y)] \\ &= I(x+1,y) + I(x-1,y) + I(x,y+1) + I(x,y-1) - 4I(x,y)\end{aligned} \qquad (7.17)$$

As máscaras definidas a partir da aproximação da equação (7.17) são mostradas na Figura 7.11.

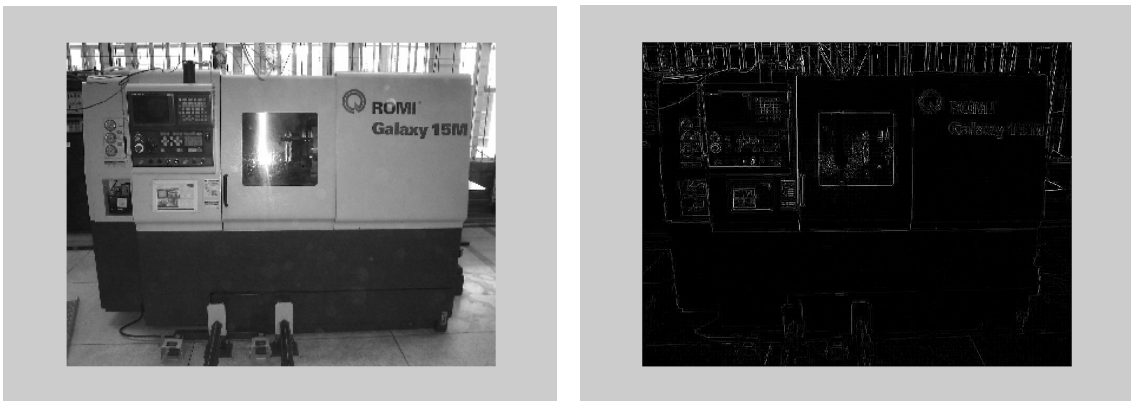
a)

b)

Figura 7.11 – Operador Laplaciano. (a) Máscaras do Operador Laplaciano, (b) Imagem filtrada utilizando o operador Laplaciano.

7.4.2.4 - Filtragem no Domínio da Freqüência

A transformada de Fourier está ligada diretamente com os chamados filtros no domínio da freqüência, através deles pode-se obter uma filtragem eficiente das imagens adquiridas. Esta filtragem é utilizada quando a imagem apresenta ruído regular e / ou periódico. Essa regularidade ou periodicidade vai se traduzir pela presença no espectro de Fourier de um pico de freqüência especifica. A remoção desse pico do espectro é feita pela transformada de Fourier inversa e portanto o ruído regular e /ou periódico.

A transformada discreta de Fourier da imagem $I(x,y)$ pode ser calculada tomando-se as transformadas de Fourier $TF(I)\,e\,TF(F)$. A transformada discreta de Fourier da imagem $I(x,y)$ é:

$$\begin{aligned} I(u,v) &= TF[I(x,y)] \\ &= \frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} I(m,n)e^{-2i\pi(\frac{mu}{M}+\frac{nv}{N})} \end{aligned} \quad (7.18)$$

A transformada de Fourier está relacionada com a convolução. A transformada de Fourier da convolução $R(x,y)$ das duas imagens $I(x,y)$ e $F(x,y)$ é igual ao produto das transformadas de Fourier de $I(x,y)$ e $F(x,y)$ como e indicado na Equação (7.19) (Facon, 2002).

$$TF(R) = TF(I(x,y) ** F(x,y)) = TF(I)TF(F) \quad (7.19)$$

A convolução da imagem I é então a transformada inversa do produto, como se mostra na Equação (7.20).

$$R(x,y) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} R(u,v)e^{-2i\pi(\frac{xmu}{M}+\frac{ynv}{N})} \quad (7.20)$$

Os filtros podem ser divididos em duas classes: filtros passa-baixa e filtros passa-alta.

7.4.3 - Segmentação de Imagens

A segmentação de imagens é um dos processos mais importantes na análise de processamento de imagens cujo objetivo principal é dividir uma imagem em seu domínio espacial em subconjuntos mutuamente exclusivos, chamados de regiões, onde cada uma destas é uniforme e homogênea com respeito a algumas propriedades, como tom ou textura.
Segundo (Facon, 2002) o alvo da segmentação é obter, a partir de uma imagem digitalizada pré-processada, um conjunto de “primitivas ou segmentos significativos “que contém a informação semântica relativa à imagem de origem.

Os métodos de segmentação podem ser divididos em três grupos: (1) O primeiro é o de conhecimento global (*global knowledge*) da imagem ou de suas partes; o conhecimento é representado usualmente pelo histograma das características da imagem. O segundo grupo é a segmentação baseada nas bordar da imagem (*Edge-based segmentation*) e o terceiro grupo é a segmentação baseada em regiões (Sonka, et al., 1999). Grande variedade de características podem ser usadas na detecção de bordas e regiões na imagem, como brilho, textura, campo de velocidades, etc.

7.4.3.1 - Segmentação por Região

A segmentação por região é o grupo que será explicado com maior detalhe por ser uma das técnicas utilizadas no processamento da imagem capturada pelo robô móvel deste trabalho, e assim poder extrair características importantes da imagem as quais serão usadas na navegação do robô.

A detecção de regiões numa imagem pode ser feita, principalmente, com um dos dois objetivos: (1) extrair uma determinada região e (2) dividir (particionar) a imagem num conjunto de regiões disjuntas cuja união representa a imagem inteira.

A região numa imagem é um conjunto de pontos "ligados" onde, de qualquer ponto da região pode-se chegar a qualquer outro ponto por um caminho inteiramente contido nessa região. As regiões que se deseja detectar, em geral, são regiões "homogêneas", ou seja, representam alguma propriedade local aproximadamente constante. Comumente esta propriedade é a continuidade no nível de cinza.

Um exemplo de função exprimindo essa propriedade de continuidade sobre uma região $R(x, y)$ pode ser (Sonka, 1999):

- $R(x, y)$ é homogênea se e somente se, para qualquer dupla de pontos, a diferença entre os níveis de cinza dos dois pontos não for maior que uma constante.

Os diferentes métodos para particionamento em regiões são:

i. Limiarização ou binarização (*Threshold*)

ii. Divisão e fusão (*Split and Merge*)

Limiarização

O principio da limiarização (Gonzalez et al., 2000) consiste em separar as regiões de uma imagem quando esta apresenta duas classes (o fundo e o objeto). Esta técnica usada no pré-processamento das imagens é baseada na utilização de um histograma. No caso de níveis de cinza bem repartidos, dado que o histograma apresenta distintamente as duas classes na forma de dois picos separados por um "vale", a limiarização é trivial.

Limiarizar consiste, no caso de um objeto mais claro que o fundo, em escolher no histograma um valor T, tal que para um pixel $P(x, y)$:

$$\begin{aligned} P(x,y) = V = 1 \quad para \quad f(x,y) \geq T \\ = 0 \quad para \quad f(x,y) < T \end{aligned} \tag{7.21}$$

onde T é o valor do limiar ou "threshold", $P(x, y) = 1$ para os elementos da imagem que representam os objetos, e $P(x, y) = 0$ para os elementos da imagem que representam o fundo (ou vice-versa) (Sonka, 1999).

Este processo é usado muitas vezes para dissociar o fundo das identidades presentes. A maior vantagem é que o principio é simples e precisa de pouca memória, mas a escolha do limiar é delicada e fica muito empírica. Ela depende muito da qualidade dos contrastes e da qualidade e da natureza da imagem. A qualidade da imagem é função do fator de refletância e do fator de iluminação (Parker, 1991). Esse último influencia fortemente a natureza dos picos do histograma. No caso de uma iluminação não uniforme sobre toda a cena, a separação dos picos e o estabelecimento do limiar são mais difíceis.

Geralmente podem ser utilizados dois tipos de limiarização (Tourino, 2002):

i. Global, quando T depende apenas do valor de $f(x, y)$

ii. Dinâmico, quando T depende de $f(x, y)$ e de $P(x, y)$

A limiarização global simples realiza o particionamento do histograma da imagem em duas regiões através de um limiar único T (objeto e fundo). A imagem é então varrida, atualizando os valores de nível de cinza de cada pixel com base na equação (7.21).
A escolha do limiar T pode ser baseada em diversos critérios, entre os quais:

- Um valor T subjetivo definido pelo usuário do sistema
- No número de pixeis desejados para a imagem resultante da limiarização
- Numa distribuição de probabilidade P que separe os objetos do fundo.

Uma abordagem mais eficiente de limiarização é obtida através da definição do limiar ótimo (Parker, 1996) que utiliza conceitos probabilísticos da estrutura da imagem para o cálculo do valor de V (ver equação 7.20).

7.4.4 - Transformação de Perspectiva (*Inverse Perspective Mapping - IPM*)

7.4.4.1 - Mapeamento Inverso de Perspectiva (*IPM*)

O ângulo de visão baixo o qual a cena é adquirida e a distancia dos objetos até a câmera são os principais aspectos que originam os chamados de efeitos de perspectiva. Estes aspectos, basicamente geométricos, contribuem para associar a diferente informação que está contida em cada pixel da imagem adquirida. O efeito de perspectiva deve ser tomado em conta dependendo do peso de informação relevante contida em cada pixel.

A solução ao problema de remoção dos efeitos de perspectiva, da imagem adquirida, consiste basicamente numa transformação geométrica conhecida como mapeamento inverso de perspectiva (*Inverse Perspective Mapping* - *IPM*). Uma vez aplicada esta técnica de transformação geométrica a imagem é re-mapeada em uma nova imagem de duas dimensões, na qual a informação contida é homogeneamente distribuída entre todos os pixeis, permitindo assim seguir com o processamento seguinte da imagem. Obviamente a aplicação desta transformação (*IPM*) requer da aquisição de condições específicas como: posição da câmera, orientação da câmera, parâmetros óticos da câmera, etc) e algumas suposições na cena representadas na imagem (definidas como conhecimento prévio). Assim a transformação *IPM* pode ser usada em ambientes estruturados, onde, por exemplo, a câmera é montada numa posição fixa ou em situações onde a calibração do sistema e os ambientes adjacentes podem ser registrados através de outro tipo de sensores (Bertozzi, 1998).

Conforme Huh (2004), *IPM* é a transformação inversa desde o plano 2D de uma imagem ao plano de coordenadas globais 3D, ou "mundo", removendo os efeitos de perspectiva.

Remoção do Efeito de Perspectiva

O mapeamento inverso de perspectiva permite transformar a imagem desde uma determinada perspectiva, mostrada na tela, a uma vista desde o céu.

O procedimento seguido para remover o efeito de perspectiva substitui a imagem entrante, re-mapeando cada pixel colocando-os em diferentes posições e produzindo uma nova matriz 2D de pixéis. A imagem resultante é uma vista superior "*top view*" da região do caminho seguido pelo veículo móvel, como se fosse observado desde uma determinada altura. Assim este processo (*IPM*), (Lee, et al., 2001) e (Lee, et al. 2002), faz um re-mapeado da imagem entrante no espaço da imagem, $I = \{(u, v)\} \in E^2$ a um novo mapa no plano $z = 0$ do espaço real (*real world space*), $W = \{(x, y, z)\} \in E^3$. A Figura 7.12 ilustra as relações entre os dois espaços. Em particular é apresentado o ponto P, do espaço real, projetado no espaço da imagem no ponto Q.

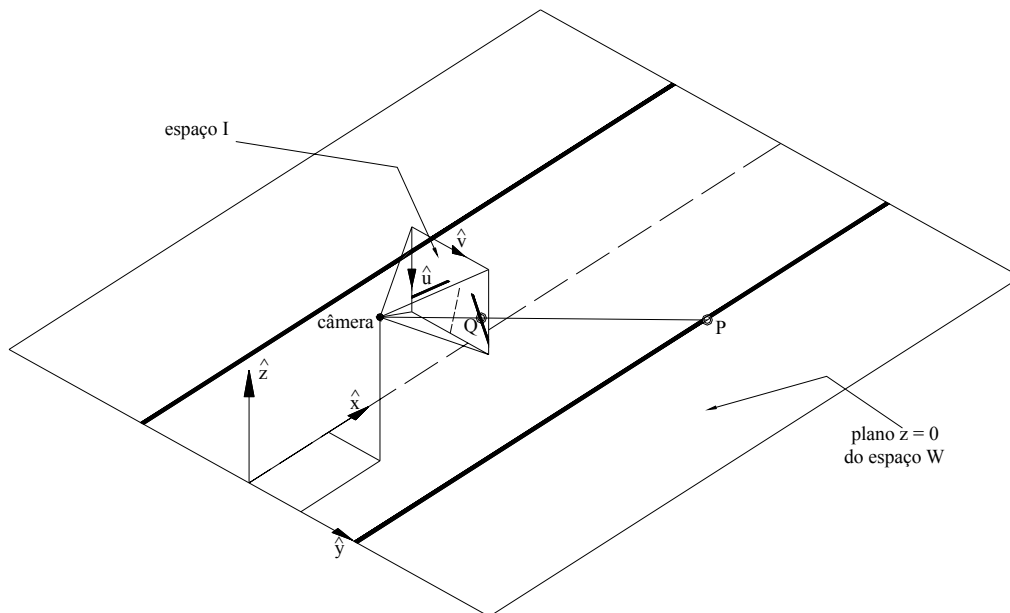



Figura 7.12 – Relações entre o espaço real (W) e o espaço da imagem (I)

Os espaços Euclidianos são definidos como:

- $W = \{(x, y, z)\} \in E^3$ representa o espaço real 3D (sistemas de coordenadas reais), onde o espaço real (real world) é definido.
- $I = \{(u, v)\} \in E^2$ representa o espaço da imagem 2D (sistema de coordenadas na tela), onde a cena em 3D é projetada.

Assumindo que o caminho é plano, a imagem adquirida pela câmera corresponde ao espaço I, enquanto a imagem reorganizada é definida no plano $z = 0$ do espaço W. Assim o processo de re-mapeado projeta a imagem adquirida no plano $z = 0$ do espaço 3D W. A imagem re-mapeada é definida como o plano xy do espaço W que será definida como a superfície $S \underline{\underline{\Delta}} \{(x, y, 0) \in W\}$

Mapeamento da Imagem no espaço I na Superfície S do espaço W

O uso da *IPM* requer do conhecimento dos seguintes parâmetros (Bertozzi, 1998):

- ponto de visão (*viewpoint*): A posição da câmera é $C = (l, d, h) \in W$;
- direção do ponto de visão (*viewing direction*): O eixo ótico $\hat{o}$ é determinado pelos seguintes ângulos:
- $\bar{\gamma}$: É o ângulo formado pela projeção do eixo ótico $\hat{o}$ no plano $z = 0$ e o eixo x. Este ângulo é definido pelo vetor unitário $\hat{\eta}$ como se mostra na Figura 7.13 (a)
- $\bar{\theta}$: É o ângulo formado pelo eixo ótico $\hat{o}$ e o vetor unitário $\hat{\eta}$ como se mostra na Figura 7.13 (b)
- abertura: A abertura angular da câmera é 2α
- resolução: A resolução da câmera é $n \ x \ n$

As equações, obtidas através de operações algébricas e trigonométricas, que definem o mapeamento final $f : I \rightarrow S$ em função de u e v são:

$$\begin{cases} x(u,v) = h\,x\cot\left[\left(\bar{\theta}-\alpha\right)+u\dfrac{2\alpha}{n-1}\right]x\cos\left[(\bar{\gamma}-\alpha)+v\dfrac{2\alpha}{n-1}\right]+l \\ y(u,v) = h\,x\cot\left[(\bar{\theta}-\alpha)+u\dfrac{2\alpha}{n-1}\right]x\sin\left[(\bar{\gamma}-\alpha)+v\dfrac{2\alpha}{n-1}\right]+d \\ z(u,v) = 0 \end{cases} \tag{7.22}$$

A Equação (7.22) retorna o ponto $(x, y, 0) \in S$ correspondente ao ponto $(u,v) \in I$

Mapeamento da Imagem na Superfície S no Espaço I

As equações que definem a transformação inversa $g : S \rightarrow I$ são (Bertozzi, 1998):

$$u(x,y,0) = \frac{\arctan\left\{\dfrac{h\,x\sin\left[\arctan\left(\dfrac{y-d}{x-l}\right)\right]}{y-d}\right\}-(\bar{\theta}-\alpha)}{\dfrac{2\alpha}{n-1}}$$

$$(7.23)$$

$$v(x,y,0) = \frac{\arctan\left[\dfrac{y-d}{x-l}\right]-\left(\bar{\gamma}-\alpha\right)}{\dfrac{2\alpha}{n-1}}$$

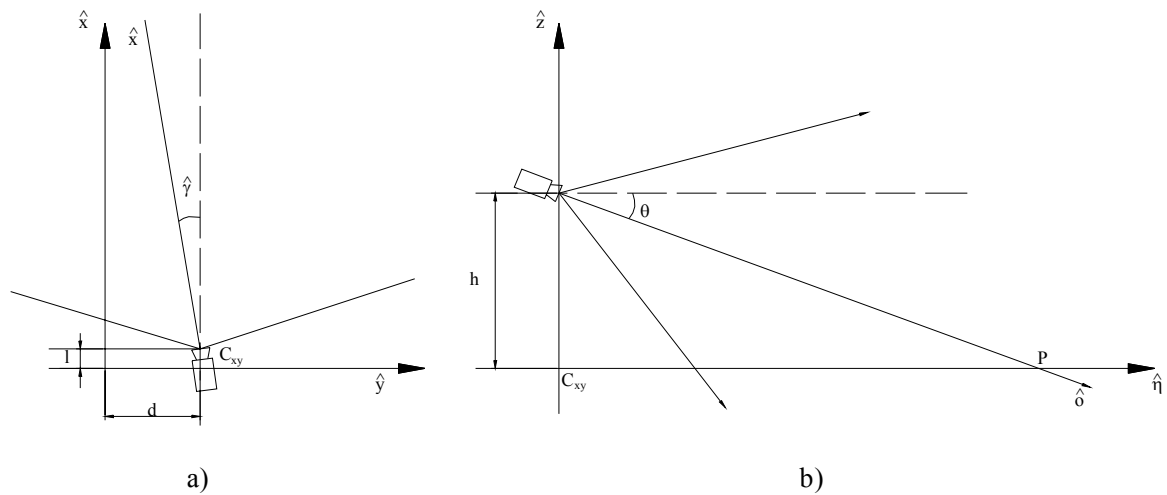



Figura 7.13 – Configurações dos parâmetros da câmera. (a) plano xy no espaço W (superfície S) (b) plano $z\eta$ assumindo que o origem é trasladado sobre a projeção C_{xy} de C em S (modificado - Bertozzi, 1998).

O uso das equações obtidas em (7.23) permite remover o efeito de perspectiva e recuperar a textura da superfície S no plano $z = 0$ do espaço real W.

A matriz de pixéis de coordenadas $(x, y, 0) \in W$, os quais formam a imagem re-mapeada, é varrida, e para cada pixel é designado o valor do pixel correspondente de coordenadas $[u(x, y, 0),\ v(x, y, 0)] \in I$.

7.4.5 - Configurações do Sensor de Visão

A configuração do sensor de visão é composta das especificações da câmera tal como o comprimento local, resolução, tamanho do pixel, etc. e geometria de posicionamento da câmera tal como altura, inclinação angular, etc.

7.4.5.1 - Campo de visão (*Field of View* – FOV)

O campo de visão (*FOV*) do sensor de visão, geralmente, pode ser determinado baseando-se na geometria da câmera, como se ilustra na Figura 7.14. O *FOV* horizontal do sensor de visão é relacionado aos parâmetros de largura, w_1 e w_2, assim como, a *FOV* vertical de-

pende das distancias d_1 e d_2. Com base nestes parâmetros de largura e distancias podem ser escritas as seguintes equações:

$$
\begin{aligned}
& d_v(k) = h_c \; x \tan\left[\left(\frac{\pi}{2} - \beta\right) - \left(\frac{\alpha_v}{2} - k\frac{\alpha_v}{n_v}\right)\right] \\
& w_v(k) = 2\sqrt{d_2(k) + h_c^2}\,.\tan\left(\frac{\alpha h}{2}\right), \quad onde \;\; k = 1 \sim n_v \\
& \Rightarrow \begin{cases} d_v(k) = d_1, w_v(k) = w_1, & quando \;\; k = 0 \\ d_v(k) = d_2, w_v(k) = w_2, & quando \;\; k = n_v \end{cases}
\end{aligned}
\tag{7.24}
$$

onde:

h_c: altura da câmera

β: ângulo da câmera

n_v: número de pixéis verticais

n_h: número de pixéis horizontais

Os ângulos horizontal e vertical da câmera, α_h e α_v respectivamente, podem ser obtidos através das equações mostradas em (7.25)

$$
\begin{aligned}
\alpha_h &= 2.\tan^{-1}\left(\frac{hCCD \, x \, n_h}{2}.\frac{1}{f}\right) \\
\alpha_v &= 2.\tan^{-1}\left(\frac{vCCD \, x \, n_v}{2}.\frac{1}{f}\right)
\end{aligned}
\tag{7.25}
$$

onde:

$hCCD$: tamanho horizontal do pixel da câmera

$vCCD$: tamanho vertical do pixel da câmera

f: distância focal da câmera dado pelo fabricante

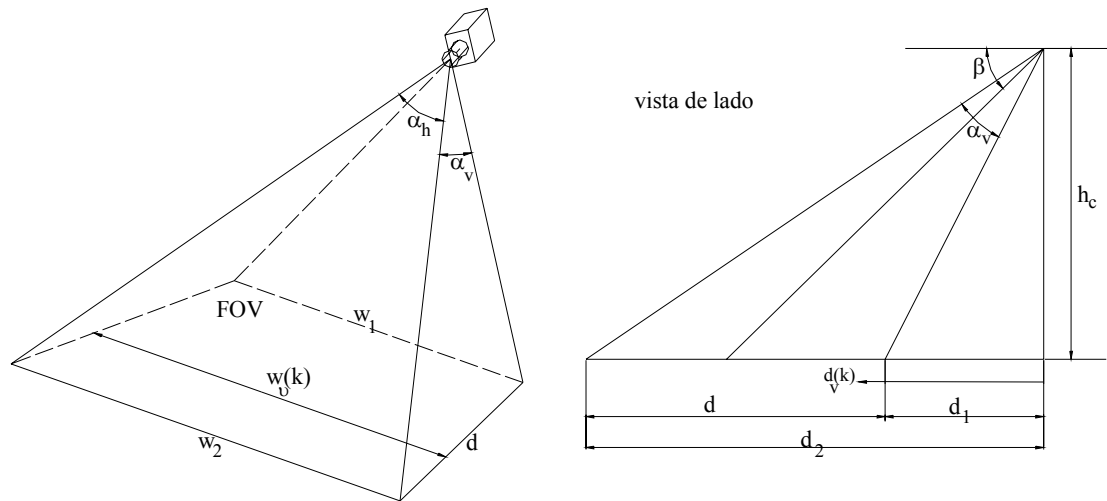



Figura 7.14 – Campo de visão definido pela geometria da câmera
(modificado - Huh, 2005)

Com base nas configurações acima descritas, do sensor de visão, pode-se dividir estes parâmetros em duas categorias: 1) parâmetros intrínsecos e 2) parâmetros extrínsecos.

- Parâmetros Intrínsecos: Abertura angular vertical (α_h) da câmera, abertura horizontal (α_v) da câmera e resolução da câmera $(n\,x\,n)$.
- Parâmetros Extrínsecos: Ponto de visão da câmera $C = (l, d, h)$ referentes ao espaço real, direção da visão da câmera referente aos ângulos γ e θ (ver Fig. 7.13).

7.4.5.2 - Resolução

A resolução de uma imagem capturada por uma câmera é dada pelo número de pixeis tanto horizontais como verticais. Assim, entre as especificações de uma câmera, dadas pelo fabricante, deve estar tanto a resolução da imagem horizontal como vertical, dadas em número de pixéis.

7.4.5.3 - Geometria de Projeção da Perspectiva

Um sistema de visão computacional envolve operações fundamentais de processamento de imagens como captura da imagem, de um objeto real, em três dimensões (3D) para posteri-

ormente ser projetada sobre uma vista de perspectiva em duas dimensões (2D). Devido à projeção de perspectiva, o resultado final (imagem capturada) conterá alguma deformidade causada, principalmente, pelo fator de distorção denominado ponto de fuga "*vanishing point*". Para se ter um melhor entendimento da geometria da projeção de perspectiva, uma explicação é fornecida usando o modelo "*pinhole*" de câmera apresentado na Figura 7.15. Este modelo geométrico é o mais comum, consistindo em um plano de projeção da imagem uv e um ponto "O" o centro de projeção. A distancia f é denominada distancia focal e o eixo Z é chamado de eixo óptico. Os eixos de referência da câmera, XYZ, definem a orientação de referência da câmera, sendo de fundamental importância nos sistemas de visão computacional (Muad, et al., 2004).

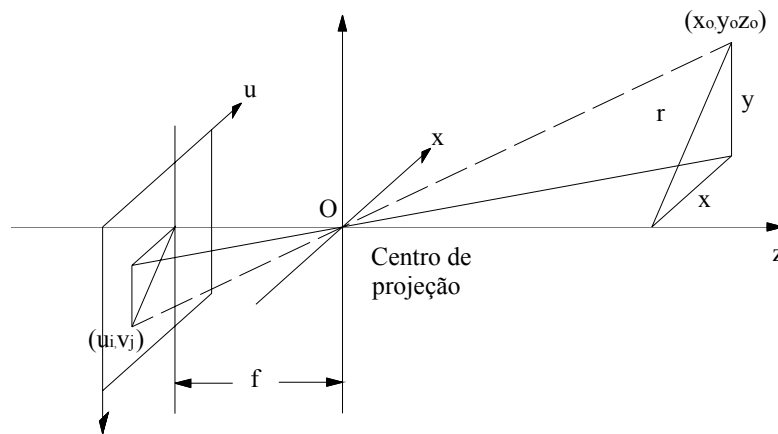



Figura 7.15 – Modelo "*pinhole*" da câmera (modificado – Muad, 2004)

A posição da imagem capturada pode ser descrita matematicamente em termos de seus componentes coordenados u e v com referência a sua posição atual no espaço em três dimensões (3D). Fazendo uso só do componente u as relações são dadas pela seguinte equação:

$$\frac{u}{x} = \frac{f}{z} \tag{7.26}$$

Agora a equação que relaciona os dois componentes u e v é:

$$(u, v) = (f\frac{x}{z}, f\frac{y}{z}) \tag{7.27}$$

O fenômeno do “ponto de fuga” ou “*vanishing point*” (Park, 2003), ocorre quando na imagem capturada existem linhas paralelas as quais ao serem projetadas no plano 2D aparecem convergindo num ponto de corte. A Figura 7.16 ilustra este fenômeno.

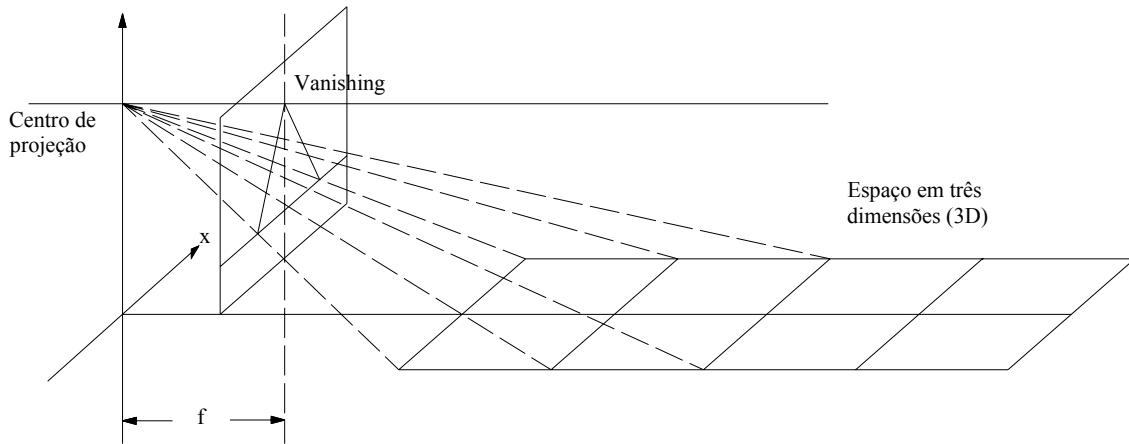



Figura 7.16 – Fenômeno “ponto de fuga” – “*vanishing point*” (modificado – Muad, 2004)

7.4.6 - Reconhecimento e Interpretação

As tarefas de reconhecimento de objetos realizadas pelos seres humanos são atos até agora misteriosos; eles percebem imagens através de sensores e transportam a informação que será processada no cérebro.

Nesta parte do trabalho será apresentada a técnica usada para o reconhecimento de objetos, baseada na forma do objeto o qual deverá ser reconhecido.
A proposta da arquitetura do robô móvel conta com um sistema de reconhecimento de formas e classificação de objetos baseados em redes neurais. O sistema de visão baseado nesta arquitetura será testado a bordo do robô móvel como suporte de sua localização e navegação em ambientes estruturados.

7.4.6.1 - Redes Neurais

As redes neurais artificiais que são utilizadas em engenharia foram inspiradas nas redes neuronais biológicas. No entanto convencionou-se chamar redes neurais artificiais a toda topologia de processamento de sinais constituída de vários elementos processadores sim-

ples altamente interconectados, i.é, estruturas baseadas no conexionismo (Bauchspiess, 2004).

A vantagem de usar redes neurais, como são comumente chamadas, não é porque as soluções que elas provem sejam elegantes ou muito rápidas; é porque o sistema “aprende” seu próprio algoritmo para a classificação de tarefas. O programa de uma rede neural básica que reconhece dígitos pode ser usada também para reconhecer distintas formas básicas como quadrados, círculos e triângulos (Parker, 1996)

7.4.6.2 - Rede Neural Básica

Uma rede neural é um conjunto de elementos de processamento interconectados (*processing elements – PE’s*), cada um dos quais realiza um cálculo muito simples. Um elemento de processamento simples, de uma rede neural, é mostrado na Figura 7.17, assim como também as entradas, o valor dos pesos para cada entrada e os valores de saída, os quais podem ser usados como entradas para outros elementos (Hayin, 1999). Cada um destes valores é numérico. O valor associado com qualquer dos nodos é chamado de ativação, o qual é simplesmente a soma de cada valor da entrada multiplicado por seu respectivo valor do peso. A saída dos *PE’s* poderiam ser simplesmente o valor de ativação, mas isto na maioria dos casos é uma função de ativação, conhecida também como função de saída.

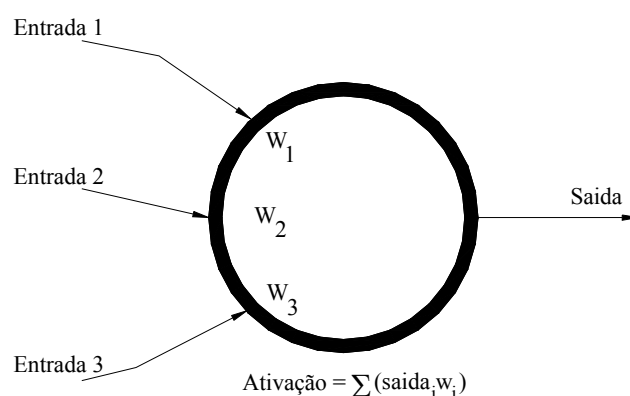



Figura 7.17 – Rede neural básica

Uma rede neural artificial é formada pela conexão de neurônios, ou nós, organizados em camadas, como se ilustra na Figura 7.18. As entradas na primeira camada (camada de en-

tradas) são recebidas pelos neurônios da camada de entrada. As saídas destes neurônios são direcionadas para as entradas dos neurônios da camada seguinte até a saída da rede.

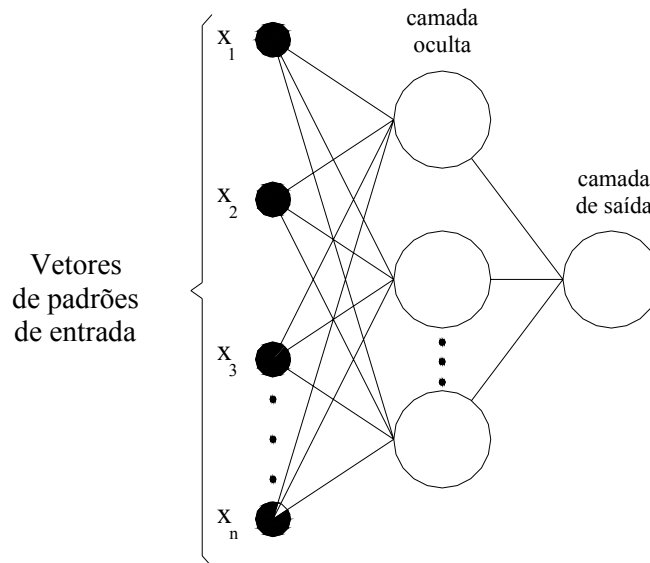



Figura 7.18 – Estrutura de uma rede neural

7.4.6.3 - Perceptron Multicamadas (MLP)

Nesta seção serão discutidos alguns conceitos básicos das redes neurais de múltiplas camadas as quais serão utilizadas para o reconhecimento de marcas pelo robô móvel durante sua navegação . O modelo escolhido conhecido como perceptron multicamada, do inglês "*Multilayer perceptrons*" (*MLP*), atuando como classificador e auto-associador será brevemente tratado nesta parte do trabalho.

O perceptron multicamada $\mathcal{N}$ é uma rede neural particular cujos neurônios são organizados em diferentes camadas. As camadas serão denotadas através da seguinte nomenclatura $l \in [0,....L]$. A camada de entrada (input layer) $(l = 0)$ recebe uma estimulação externa (forma, padrão) já a camada de saída (output layer) $(l = L)$ dá a resposta do trabalho feito pela rede (Adorni, et al., 1999). Cada neurônio na camada genérica l pode receber entradas somente dos neurônios pertencentes à camada l' com $(l' < l)$. O número de neurônios por camada é denotado por $n(l)$. Cada neurônio de uma camada l é representado por $i(l) = 1,\ldots,n(l)$.

Quando um estímulo S é dado como uma entrada à rede $\mathcal{N}$, se terá para cada neurônio $i(l)$ uma ativação $a_{i(l)}(s)$ e sua saída $o_{i(l)}(s)$ respectiva. A ativação do neurônio é computada através da propagação adiantada das saídas dos níveis de neurônios prévios pela equação (7.28).

$$a_{i(l)}(s) = w_{i(l)} + \sum_{j(l-1)=1}^{n(l-1)} w_{i(l).j(l-1)}.o_{j(l-1)}(s) \tag{7.28}$$

onde $w_{i(l).j(l-1)}$ é denotado como o peso da união entre os neurônios $j(l-1)$ e $i(l)$, e $w(l)$ é o limiar de ativação. A saída do neurônio $i(l)$ é relacionado a sua ativação através da equação (7.29).

$$o_{i(l)}(s) = f(a_{i(l)}(s)) \tag{7.29}$$

onde $f : \mathcal{R} \to [0,1]$ é uma função contínua "*squashing-like*" com uma primeira derivada positiva contínua. Em sua forma geral uma rede $\mathcal{N}$ é definida pela equação.

$$\mathcal{N}: \mathcal{R}^{|n(0)|} \to \mathcal{R}^{|n(L)|} \tag{7.30}$$

onde $|n(0)|$ é o número de entradas, e $|n(L)|$ é o número de saída dos neurônios. Os exemplos usados na fase de aprendizagem são basicamente uma coleção de P pares de entradas/saídas que a rede $\mathcal{N}$ deve interpolar, assim:

$$\mathcal{L}_e = \{ (I(s), D(s)) : I(s) \in \mathcal{R}^{n(0)}, D(s) \in \mathcal{R}^{n(L)},\ s = 1, \ldots, P \} \tag{7.31}$$

A função de custo para um certo $\mathcal{L}_e$ é mostrada na equação (7.32) :

$$E_P \cong \frac{1}{2} \sum_{s=1}^{P} E_s = \frac{1}{2} \sum_{s=1}^{P} \sum_{j=1}^{n(L)} \left[d_{j(l)}(s) - o_{j(L)}(s) \right]^2 \tag{7.32}$$

A Equação (7.32) é otimizada durante a fase de aprendizagem à medida que são produzidos os valores dos pesos. Segundo Adorni (1999), este processo de otimização poderia ser severamente propagado, ou generalizado, pela presença do local mínimo existente no trei-

namento da rede através da técnica de *backpropagation* especialmente quando a rede trabalha com mapeamentos complexos.

7.4.6.4 - Treinamento da Rede

Os pesos de uma rede neural já treinada não variam. O treinamento da rede neural é um processo, onde os pesos variam com o tempo até conseguir os pesos finais ao final do treinamento. Uma rede neural treinada adquire um "conhecimento" o qual é refletido nos pesos que são aplicados às entradas de seus neurônios.

O processo de aprendizagem é feito em épocas. Cada época consiste na apresentação de todas as amostras disponíveis para o treinamento. No modo de aprendizagem seqüencial, devem-se encontrar os erros e realizar a adaptação dos pesos. Ao final da época, isto é, quando todas as amostras tiverem sido apresentadas, verifica-se o erro quadrado médio de todas as saídas, calculado pela equação (7.32). Caso este erro seja igual ou inferior ao erro mínimo esperado o treinamento chegará ao fim. Se o erro mínimo, não for alcançado deve-se iniciar uma nova época, onde todas as amostras serão apresentadas novamente em uma ordem aleatória diferente da utilizada na época anterior.

7.4.6.5 - O Algoritmo Backpropagation

O algoritmo *Error Backpropagation* é em geral muito lento e suscetível a "patologias" de treinamento. A paralisia da rede ocorre em regiões de gradientes próximos de zero. Dependendo das condições iniciais o treinamento fica preso em mínimos locais da superfície de erro (Bauchspiess, 2004).

O algoritmo *Error Backpropagation* levou a uma grande aceitação das *RNA's* por parte da comunidade científica, uma vez que redes multicamadas podem ser treinadas a partir dos dados que representam amostras significativas do processo em questão.

8 – CARACTERÍSTICAS DO HARDWARE E SOFTWARE DO ROBÔ MÓVEL

8.1 – ROBÔ MÓVEL

O robô móvel *Nomad XR4000*, utilizado no desenvolvimento deste projeto, é um sistema de robótica móvel avançado cujos componentes principais compreendem sistemas de controle, comunicação, sensoriamento, administração de potência e tecnologias de programação. Este robô possui outras capacidades para o desenvolvimento de pesquisas mais avançadas como são os casos de manipulação móvel, movimentos com velocidades altas, desenvolvimento de sistemas inteligentes utilizando o sistema de visão e navegação baseada no sistema sensorial. A Figura 8.1 mostra o robô móvel utilizado (Nomadic Technologies, 1999).

Figura 8.1 – Robô móvel - Nomad XR4000

8.1.1 – Descrição do Hardware do Robô

O *hardware*, conjunto de componentes que conformam a parte física do robô móvel, é composto pelos subsistemas de movimentação, sensorial e de comunicação (Nomadic Technologies, 1999).

8.1.1.1 - Subsistema de movimentação

O sistema motor do robô móvel, composto de 8 motores e 4 rodas, baseia-se num sistema holonômico, provendo três graus de liberdade (X, Y, θ) : dois de translação e um de rotação. Suas rodas possuem eixos de translação y de rotação independentes, somando oito motores para as quatro rodas do sistema. Três *DSP* (*Digital Signal Processing*) e um microcontrolador realizam o controle dos oito eixos e o cálculo da estimativa da posição (*dead reckoning*), que é baseado no modelo cinemático do robô e em dados provenientes de encoders localizados nos atuadores do robô.

8.1.1.2 - Subsistema Sensorial

- **Sensores de ultra-som**: o sistema de sensores ultrasônicos, sistema *Sensus* 250, consiste em dois anéis contendo 24 sensores, provendo informação sobre a distância de objetos entre 150 e 7000mm. Esses dados são calculados através do conhecimento da velocidade do som e do tempo de viagem do pulso emitido e refletido. A equação (8.1) mostra como calcular a distância do objeto.

$$d_{objeto} = v_{som} . t_{viagem} \tag{8.1}$$

  O sonar baseia-se no emissor ultrasônico *Polaroid* 6500, que emite 16 ciclos de onda quadrada de 49,4 kHz através de um transdutor eletrostático, que também funciona como receptor após o disparo do pulso.

- **Sensores Infravermelhos**: O sistema *Sensus* 350 consiste em dois anéis contendo 24 sensores por luz infravermelha capazes de prover informações de presença de objetos próximos a distâncias entre 300 e 500mm.

  Estes sensores são compostos de dois emissores infravermelhos *SIEMENS SFH* 34 – 3 *Ga* – *As* e um receptor *SIEMENS SFH* 2030*F* (fotodiodo). Para obter a energia refletida por um objeto, são feitas duas leituras: uma com os emissores desligados – obtendo-se o nível de radiação ambiente, e uma com os emissores ligados obtendo-se a energia refletida mais

a radiação ambiental. A energia referente ao objeto é obtida da subtração das leituras obtidas. A equação (8.2) mostra o cálculo desta energia.

$$E_{objeto} = E_{ligado} - E_{desligado} \qquad (8.2)$$

Os fatores que influenciam as medições obtidas pelos sensores são os seguintes: geometria (quanto maior a distância do objeto, menor será a energia refletida captada pelo receptor), iluminação (quantidades grandes de energia infravermelha no ambiente, pode saturar o receptor, reduzindo sua sensitividade e cor de superfície. A refletividade do objeto à luz infravermelha, a cor, possui grande influencia nas leituras obtidas.

8.1.1.3 - Subsistema de Visão

O robô móvel além do subsistema sensorial possui o sistema *Sensus* 460, que consiste numa câmera de vídeo colorida *Hitachi KP – D*50 *color digital*, padrão *NTSC* (*National Television Standards Comittee*), e uma placa de captura de vídeo, *Matrox Meteor*, padrão *PCI* (Peripherical Component Interconnect), com até 45 *Mbytes / s* de transferência de dados. Como complemento o subsistema de visão é auxiliado por uma unidade "*Pan – Tilt*", a qual é controlada por uma porta serial RS-232, para movimentação da câmera.

8.1.1.4 - Subsistema de Comunicação

A comunicação do robô móvel com a rede local é realizada através da interface Pr*oxim RangeLan* 2 (freqüência de 2.4 GHz.), e um rádio *Ethernet*. Esse sistema fornece acesso à rede local utilizando o protocolo *TCP/IP*.

8.1.2 – Arquitetura XRDEV (Nomadic Technologies, 1999)

XRDEV é uma arquitetura multi-processada. Uma aplicação XRDEV é constituída por vários processos: processos do robô, processos do usuário, processos da interface, etc. comunicando-se através da rede. Não há limite para o numero de robôs que podem ser controlados na arquitetura XRDEV nem para o número de programas do usuário que controla o robô. Também não há restrições no número de robôs que podem ser controlados por um

programa do usuário nem no controle compartilhado de um robô por vários programas do usuário. Os programas dos usuários podem ser executados de qualquer lugar na rede. Estes processos existentes são:

- **Processo** *NRobot* : É o processo que controla o robô, comunicando-se diretamente com o hardware do robô. Pode existir quaisquer números de processos *NRobot* na aplicação, mas somente um processo *NRobot* sendo executado em cada robô.
- **Processo** *Ngui* : É o processo que oferece uma interface gráfica. Através dessa interface, um usuário pode enviar comandos e receber dados de qualquer processo do robô.
- **Processo do usuário**. É o programa do usuário que controla o robô.

Há algumas configurações possíveis utilizando-se varias combinações desses processos. A configuração mais simples é o *NRobot* sendo executado no robô e a *Ngui* sendo executada em outra máquina, comunicando-se pelo radio Ethernet. O usuário controla o robô manualmente pela *Ngui* . A Figura 8.2 (a) mostra o diagrama dessa configuração.

(a) (b)

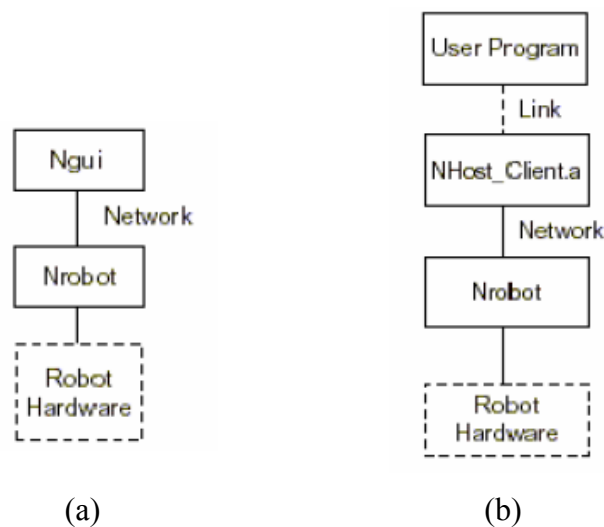



Figura 8.2 – Configurações da arquitetura XRDEV do robô móvel (a) Configuração simples – um *GUI* , um robô (b) Configuração simples via rede. (Nomadic Technologies, 1999)

Uma configuração mais elaborada envolve um processo *NRobot* e um programa cliente usuário ligando à biblioteca cliente-hóspede (*Host Client Library*). Nessa configuração, o

programa do usuário com "*link*" na biblioteca cliente – hóspede pode ser compilado e e-xecutado em qualquer máquina que possua uma conexão ao robô. Esta biblioteca é conhecida como *Nhost_client.a* e se comunica diretamente com o processo *NRobot* e permite acessar as estruturas usando os sensores, sistemas de movimentação, etc. O processo *NRobot* é executado no próprio robô. A Figura 8.2 (b) mostra o diagrama dessa configuração (Dutra, 2003).

8.1.3 – A Linguagem do Robô - Interfaces de Programação

A estrutura *N_RobotState*, onde são armazenados todos os dados dos sensores e de configuração do robô, é a base da programação do robô móvel. Essa estrutura contém toda a interface de comunicação entre o "software" e o "hardware do robô. Um ponteiro para essa estrutura é recebido pela chamada da função *N_GetRobotState*. A modificação dos parâmetros dessa estrutura indicará o comportamento dos módulos componentes do robô. O apêndice A mostra a estrutura *N_RobotState* completa utilizada no robô móvel.

O acesso a *N_RobotState* é obtido através do comando *N_GetRobotState*, cuja sintaxe é mostrada abaixo. Este comando é declarado no inicio do programa.

```
struct N_RobotState *N_GetRobotState (long RobotID);
```

A leitura dos dados da estrutura pode ser feita através da família de comandos *Get…* ou atualizar dados da estrutura com a família de comandos *Set…*.

No Apêndice A se apresentam com detalhe todas as estruturas e funções de cada um dos sistemas que formam parte do robô móvel, os quais em conjunto podem ser integrados para conseguir uma eficiente navegação. Estes sistemas são: sistema de comunicação, sistema de movimentação do robô, sistema de localização do robô, sistema da configuração integrada, sistema de sensores (colisão, infravermelhos, ultra-som) e o sistema de alimentação.

8.1.4 - Placa de Captura de Imagens

O sistema de visão do robô móvel possui uma placa de captura de vídeo *Matrox Meteor* e uma câmera *Hitachi KP – D50 Color*. A placa de captura permite taxas de atualização de até 30 fps (imagens por segundo) e resoluções de até 640 x 480 pixeis para imagens de 24 bits.

A câmera possui uma lente de 6mm 1: 1.2 com variações do foco e diafragma sem zoom. A configuração da placa de captura é realizada através da biblioteca *ioctl_meteor.h*, que define estruturas de dados e variáveis para esse propósito. A configuração da imagem capturada é feita através da estrutura *meteor_geomet* a qual possui quatro variáveis conforme a Tabela 7.1 (Tourino, 2000).

Tabela 7.1: Variáveis da estrutura de dados *meteor_geomet*

Variável	***Uso***
rows / columns	define as dimensões da imagem
frames	define o número de imagens na memória
oformat	define a quantidade de níveis de quantização de cores

Os dados de configuração são enviados para o "*hardware*" da placa de captura através da chamada da função *ioctl* do UNIX. A seqüência de configuração é dada pelas seguintes etapas: a) definição da geometria da imagem, b) definição do tipo de vídeo, c) definição do tipo de entrada, d) definição do sinal de imagem disponível e e) definição do modo de captura. No Apêndice A se apresentam as estruturas que definem esta configuração. Estas etapas são detalhadas no Apêndice B.

9. PARTE EXPERIMENTAL

9.1 – ARQUITETURA DO SISTEMA

A organização do controle da arquitetura afeta totalmente o desempenho do sistema do robô móvel. A integração de múltiplos sistemas de sensores, sem uma aquisição e pré-processamento de dados adequada e eficiente tem sido freqüentemente ineficaz. Pelo contrario, múltiplas formas de desenvolvimento e tarefas de controle podem ser mais eficientes e manipuláveis por uma arquitetura organizada provendo suporte para programação de eventos (priorizando tarefas), processos de interação e de controle (Beccari, et al., 1997).

O controle da arquitetura em tempo real, que será implementada no robô móvel, está formado por unidades de tarefas organizadas de acordo as prioridades exigidas pelo sistema. Estas unidades ou módulos são:

1. **Unidade de Captura da imagem**

   A unidade de captura é a responsável pela aquisição da imagem através da câmera, assim como também pela configuração da mesma e armazenar a imagem capturada em uma matriz utilizada nos módulos seguintes.

2. **Unidade de Pré-processamento**

   Esta unidade é responsável por realizar eficientemente as operações fundamentais nas imagens capturadas. É nesta unidade onde o ruído e outros fatores, como a luminosidade do ambiente, são atenuados permitindo obter a informação necessária da imagem utilizando os algoritmos específicos de visão. Alguns dos algoritmos utilizados nesta fase são: limiarização, gradiente, filtragem, etc.

3. **Unidade de Transformação da Perspectiva (IPM)**

   Esta unidade é responsável da transformação geométrica da imagem capturada. Aqui, os pontos mais importantes da imagem distorcida são re-mapeados em uma nova imagem de duas dimensões, na qual a informação contida é homogeneamente distribuída entre todos os pixéis, permitindo assim seguir com o processamento seguinte da imagem

4. **Unidade de Rastreamento de linhas através do sistema de visão**

   Nesta unidade é realizada a principal operação de visão requerida pelo sistema de navegação do robô móvel. Esta operação consiste em computar a direção da linha rastreada, desenhada no chão, pelo sistema de visão do robô móvel. Este processamento consiste em estimar a inclinação da linha tomando dois pontos da imagem capturada e pré-processada.

5. **Unidade de reconhecimento de sinais através do sistema de visão**

   Esta unidade é responsável de identificar sinais usando o sistema de visão do robô móvel, e sugerir ou ordenar alguma ação de movimentação a ser feita pelo robô durante sua navegação. Sinais especiais determinam o fim ou inicio de alguma ação que está sendo executada pelo robô delimitando o ambiente de navegação. Sinais simbólicos serão utilizados no processo de reconhecimento, em vez de sinais de difícil reconhecimento por humanos como é o caso de código de barras, devido ao potencial presença de pessoas no ambiente de navegação. Assim, sinais alfanuméricos poderiam ser usados, no processo de reconhecimento, durante a navegação do robô móvel. A técnica utilizada para o reconhecimento de sinais é baseada em algoritmos de redes neurais. Esta técnica é chamada "multicamadas" (MLP) treinada através do algoritmo "backpropagation" (ver seção 7.3.5).

A Figura 9.1 mostra a arquitetura do sistema composta pelos módulos mencionados anteriormente.

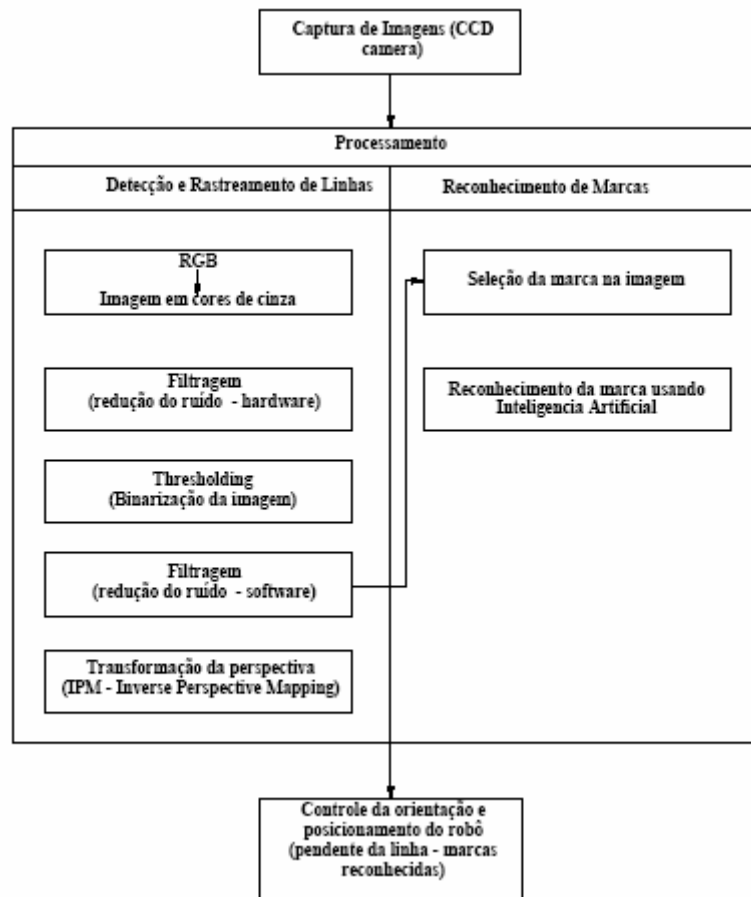



Figura 9.1 – Arquitetura do Sistema

9.2 – MODELAGEM E IMPLEMENTAÇÃO DA CÉLULA FLEXÍVEL DE MANUFATURA (GRACO – UnB)

Nesta seção serão apresentados, brevemente, todos os trabalhos desenvolvidos na implementação física e lógica da *FMC* no laboratório GRACO – UnB.

Uma das primeiras tarefas que devem ser desenvolvidas para realizar a implementação de uma *FMC* é sua modelagem. Esta tarefa implica na análise dos principais aspectos relacionados à tomada de decisões, avaliando o comportamento do sistema em diferentes possíveis estados de funcionamento. Dependendo da diversidade de peças que serão produzidas na *FMC*, estes estados serão modelados e simulados através de diferentes métodos. Predizer possíveis falhas no funcionamento da *FMC*, principalmente de posicionamento de seus componentes, é também um dos objetivos nesta parte do trabalho.

A modelagem da *FMC* será desenvolvida através das seguintes formas: 1) modelagem física e 2) modelagem lógica. A modelagem física refere-se basicamente à localização das estações de trabalho da *FMC* as quais definem o *layout* da célula. Esta modelagem pode ser feita através da utilização de "*softwares*" de simulação em 3D e assim possam ser avaliados os movimentos e trajetórias principalmente das unidades de manuseamento e transporte de material. Já a modelagem lógica implica a utilização de eventos discretos, representados pelos diferentes estados dos componentes da *FMC* durante seu funcionamento. Esta modelagem pode ser feita utilizando "Redes de Petri"; através desta técnica são modelados os diferentes estados das estações de trabalho, garantindo o controle do chão de fábrica evitando, possíveis falhas, principalmente de colisão e tempos mortos entre as estações de trabalho.

Uma das técnicas mais usadas na modelagem de sistemas de controle do chão de fábrica são as técnicas de objetos orientados. Um exemplo de modelagem utilizando estas técnicas, através da linguagem *UML (Unified Modeling Language)*, pode ser visto no trabalho apresentado por Yang (1999).

Outra técnica muito utilizada para realizar modelagens de sistemas automatizados e não automatizados é baseada nas metodologias *IDEF*. Esta técnica pode ser utilizada primeiramente para definir as necessidades e especificar as funções do sistema que se está implementando, posteriormente pode ser projetada uma implementação que reúna os requisitos e realize as funções.

A implantação da *FMC* é outro ponto que será tratado nesta parte do trabalho. Seu desenvolvimento será realizado através de duas fases: 1) integração física e 2) integração lógica. A integração física envolve todo o concernente à definição do *layout* da célula (posicionamento das estações de trabalho), assim como também à projeção e construção dos equipamentos necessários (garra do manipulador de material, sistema de deslocamento do manipulador de material, sistema de armazenamento de peças, suporte da unidade de controle de qualidade, conexão dos cabos de comunicação, etc) para o funcionamento de cada uma das estações de trabalho e total da *FMC*. A integração lógica refere-se à implementação da unidade de gerenciamento da *FMC*, e a programação tanto do manipulador como do transportador de material.

9.2.1 - Célula Flexível de Manufatura

9.2.1.1 - Componentes da Célula Flexível de Manufatura Implementada

A *FMC* que esta sendo desenvolvida no laboratório é formada pelos seguintes equipamentos ou componentes: um torno *CNC* (marca ROMI, modelo Galaxy 15M, sistema de PLC – FANUC), um braço robô (ASEA IRB6 L2E), um micrometro laser (Micrometro Laser MITUTOYO, modelo LM6100), um robô móvel (NOMAD XR4000, Fabricação da Nomadic Technologies) e um sistema de armazenamento de peças.

- **Torno CNC**: responsável pelas operações de usinagem das peças.
- **Braço robô**: trabalha como uma unidade de manuseamento e transporte de material.
- **Micrômetro laser**: trabalha como uma unidade de controle de qualidade e classificação das peças fabricadas. O controle e classificação são baseados basicamente no cumprimento das tolerâncias previamente estabelecidas pelo fabricante. Elas podem ser classificadas como: boas para re-processo e defeituosa..
- **Robô móvel**: trabalha como *AGV* e suas funções são basicamente de transporte do material entre as estações de trabalho
- **Armazém de peças**: lugar onde o material de trabalho e peças fabricadas são armazenados.

A Figura 9.2 (a) mostra os equipamentos que conformam a *FMC*, a Figura 9.2 (b) mostra as diferentes vistas de vídeo mostradas na pagina web: http://video.graco.unb.br.

.

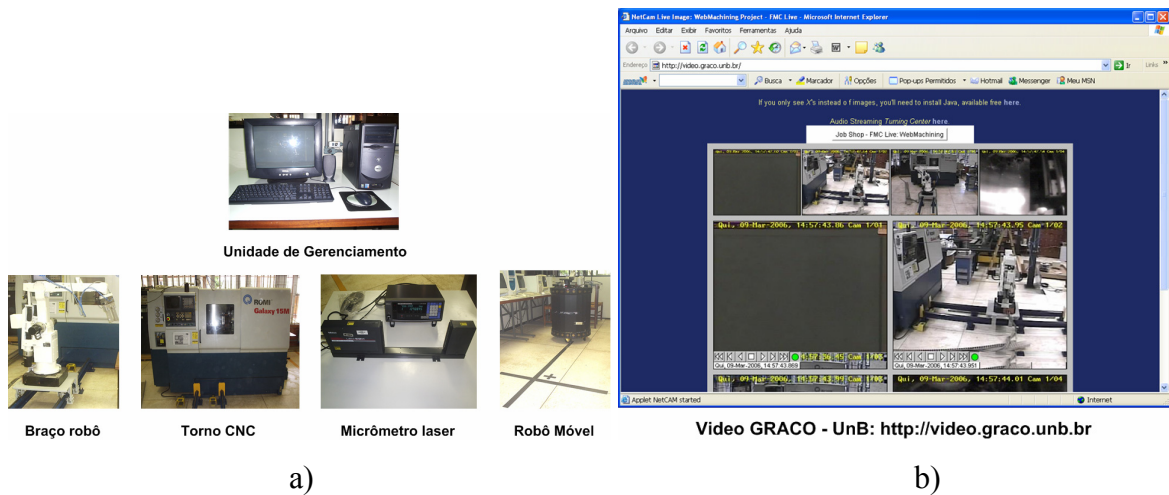



Figura 9.2 – Célula Flexível de Manufatura. (a) Componentes da FMC (b) Imagem da pagina web: http://video.graco.unb.br,

9.2.1.2 - Modelagem da Célula Flexível de Manufatura

Modelagem Física

Esta parte é muito importante no desenvolvimento futuro das funções da *FMC*. A modelagem foi feita utilizando um software de desenho 3D (*Workspace*) e tem como objetivo principal, avaliar o sincronismo dos movimentos das unidades de manuseio e transporte de material feitos pelo braço robô e o *AGV* respectivamente. Através desta ferramenta de modelagem e possível realizar simulações destes movimentos com precisão prevendo assim colisões que possam acontecer durante o funcionamento da *FMC*. O posicionamento do manipulador (braço robô) no instante de posicionar o material a ser usinado na placa do torno é uma das situações que requer muita precisão. Através da sua modelagem e simulação na *FMC* podem-se obter as coordenadas dos movimentos feitos e garantir um bom funcionamento da *FMC*. A Figura 9.3 ilustra os estados, durante o funcionamento da *FMC*, que requerem de precisão no posicionamento das unidades (Texeira, et al., 2005)

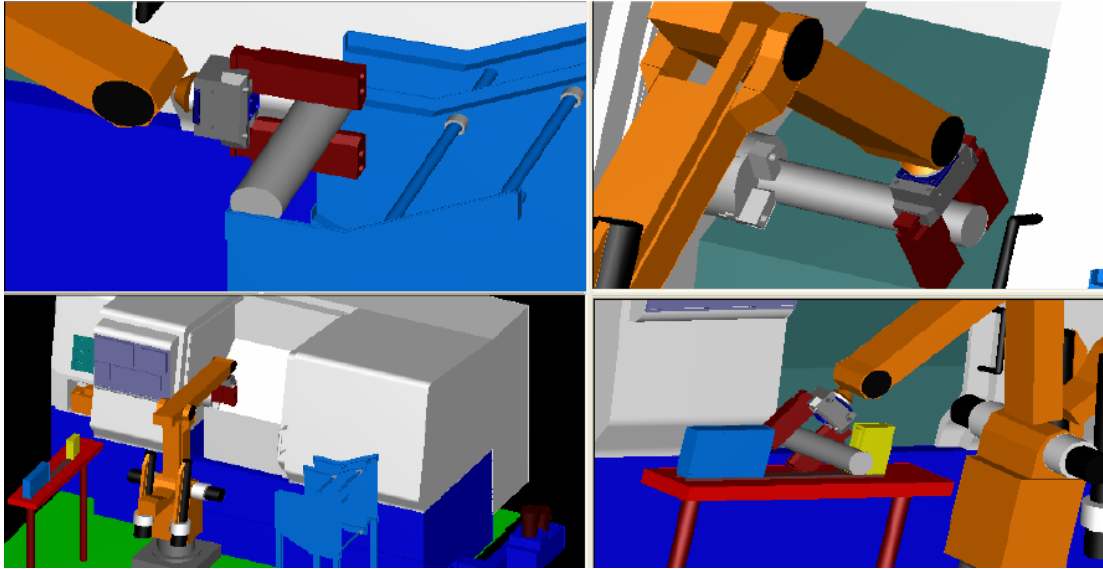



Figura 9.3 – Modelagem física dos diferentes estados, durante o funcionamento da FMC.

Modelagem Lógica

A técnica utilizada na modelagem lógica da *FMC* foi as Redes de Petri (RP). Em manufatura, são importantes as linhas de transferência, isto é, os sistemas formados por máquinas e estoques intermediários, em cascata. Com base nisto as RP's são ferramentas ideais para a realização da modelagem da *FMC*. Segundo (Couto de Moraes, 2001), as RP's destacam-se na engenharia atual pelas seguintes qualidades:

- capturam as relações de precedência e os vínculos estruturais dos sistemas reais;
- são graficamente expressivas;
- modelam conflitos e filas;
- têm fundamento matemático e prático;
- admitem varias especializações (RPs temporizadas, coloridas, estocásticas, de confiabilidade, etc.).

Uma definição básica de uma *RP* pode ser a seguinte: Uma *RP* é um grafo orientado que tem dois tipos de nós: transições e posições. Os arcos do grafo partem de algumas posições para algumas transições ou vice-versa; aos arcos associam-se números inteiros fixos, que são seus pesos. Cada posição pode conter um número inteiro de marcas, e estas, sob certas condições, podem mover-se ao longo dos arcos respeitando seus sentidos (Couto de Mora-

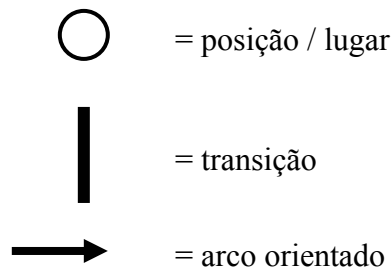
es, 2001). Matematicamente uma RP pode ser definida da seguinte forma: Uma função formada por cinco variáveis $(P, T, A, W, m0)$ em que:

- $P = \{p_1, ... p_n\}$ é um conjunto finito de posições ou lugares;
- $T = \{t_1, ... t_m\}$ é um conjunto finito de transições;
- A é um conjunto finito de arcos pertencente ao conjunto $(PxT) U (TxP)$, em que (PxT) representa o conjunto dos arcos orientados de p_1 para t_1, também designados por (p_1, t_1), e (TxP) representa o conjunto dos arcos orientados de t_1 para p_1, ou (t_1, p_1);
- W é a função que atribui um peso w (inteiro) a cada arco;
- $m0$ é um vetor cuja i-ésima coordenada define o número de marcas na posição p_n no início da evolução da rede.
- Os conjuntos T e P são disjuntos: $T \cap P = \emptyset$.
- $n = |P|$ é a cardinalidade do conjunto P, o número de posições da RP
- $m = |T|$ é o número de transições da RP.

Na representação gráfica de uma RP são utilizados os seguintes símbolos:

○ = posição / lugar

| = transição

→ = arco orientado

A Figura 9.4, mostra a modelagem dos diferentes estados durante o funcionamento da FMC usando uma RP simplificada.

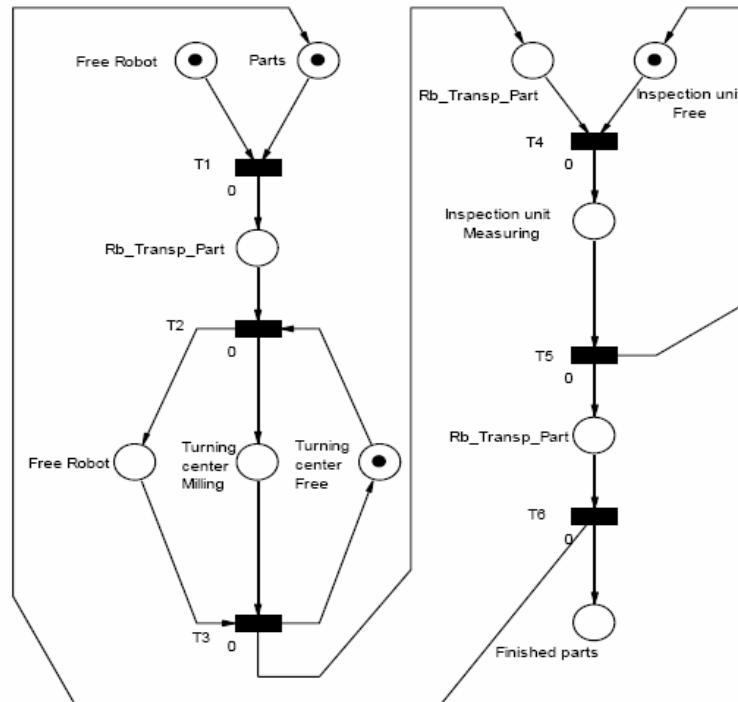



Figura 9.4 – Rede de Petri simplificada da FMC

O objetivo desta seção foi, somente, apresentar as RP como uma alternativa de modelagem existente muito usada para sistemas dinâmicos a eventos discretos como é o caso de uma FMC. Mais detalhes desta modelagem, usando RP, é apresentado no artigo *"Modeling and implementation of a Flexible Manufacturing Cell (FMC)"* (Texeira, et al., 2005).

9.2.1.3 - Integração Física

Integrar todos os equipamentos, os quais conformam a parte física de uma FMC, para que possam interagir entre si, com sincronismo, durante o funcionamento da mesma é o principal objetivo desta seção. Para conseguir esta integração são necessárias as seguintes condições:

- *layout* da FMC, localização das estações de trabalho.
- adquirir os equipamentos com as características técnicas requeridas (torno CNC, micrômetro, braço robô e robô móvel).
- conexão das interfaces de comunicação das diferentes estações
- projeto e construção do sistema de posicionamento do manipulador (braço robô)

- projeto e construção da garra para o manipulador .
- projeto e construção do armazém de partes (ver Fig. 9.7).

9.2.1.4 - Interfaces de Monitoramento e Comunicação

A comunicação entre todos os equipamentos que compõem a FMC e baseada em protocolos de comunicação. O objetivo nesta fase da implementação é chegar a comunicar todas as máquinas numa só rede de comunicação, de maneira que possam interagir entre elas tendo em comum um controlador central que é o gerenciador da FMC. Os diferentes tipos de comunicação que estão sendo implementados na FMC são: RS232, TCP/IP, I/O. Na seção 9.2.1.6 se mostrará mais detalhadamente os tipos de comunicações especificas existentes entre cada uma das máquinas da FMC.

A interface de monitoramento tem como objetivo adequar os sinais, provenientes dos controladores das estações de trabalho, para que possam ser lidas pelo servidor do gerenciador da FMC (Texeira, et al., 2005).

9.2.1.5 - Integração Lógica da FMC – Desenvolvimento de Software

A integração lógica da FMC tem como objetivo o desenvolvimento do *software* necessário para iniciar os processos de produção no chão de fábrica. Os principais softwares que estão sendo desenvolvidos estão relacionados às tarefas que serão realizadas pela unidade de transporte e manuseio de material (braço robô), a unidade de gerenciamento da FMC e o sistema de navegação do robô móvel (AGV).

- **Unidade de transporte e manuseio de material**
  O braço robô é programado, através de uma linguagem própria, para realizar deslocamentos com precisão durante o processo de produção da FMC em coordenação com as outras funções dos equipamentos da célula.
- **AGV**
  O robô móvel, trabalhando como um AGV, e programado para poder navegar pela célula realizando as seguintes funções: 1) navegação baseada no sistema de visão do robô, através do reconhecimento de linhas e marcas no chão da fábrica, 2) re-

cepção das partes fabricadas, através do uso de um dispositivo de armazenamento colocado no robô para recepcionar estas partes, e 3) Transportador de materiais (“*material handling”*) durante a manutenção dos equipamentos da célula e “*setup*” das máquinas. A robô móvel se comunica diretamente com o gerenciador da célula, que comanda as tarefas que deveram ser executadas (Villanueva. et al., 2005).

- **Unidade de Gerenciamento**

  A concepção é o desenvolvimento do controle da célula, filosoficamente falando, é baseado em dois importantes fatores (Leitão, 1995): 1) uma estrutura modular, que permita expansões futuras do controlador da célula quando a quantidade de máquinas seja incrementada, e 2) a máxima flexibilidade do controle da célula relacionada com o chão de fábrica e arquitetura de controle. Já segundo Groover (2005), o controle do chão de fábrica está relacionado com o lançamento de ordens de produção na fábrica assim como o monitoramento e controle de ditas ordens. Estas atividades de controle do chão de fábrica são estabelecidas em três módulos: 1) Escalonamento (*scheduler*), 2) envio seqüenciado de ordens (*dispatcher*) e 3) monitoramento. O desenvolvimento do *software* que funcionará como gerenciador da célula é baseado nesses três módulos referidos (Texeira, et al., 2005).

9.2.1.6 - Estrutura Hierárquica de Controle e Comunicação da FMC

A estrutura hierárquica de controle da FMC está dividida basicamente em 4 níveis: 1) ordem do cliente, 2) sistemas CAP e CAD/CAPP/CAM, 3) gerenciador da célula e 4) componentes da célula, como mostrado na Figura 9.5. As atividades da célula deverão ser iniciadas com uma ordem registrada pelo cliente no site: http://www.webfmc.com.br. Esta ordem deverá ter a seguinte informação: data, número de partes a serem produzidas, produto selecionado para ser produzido, etc. Esta informação é processada e usada pelos outros níveis hierárquicos que conformam a FMC.

O produto ou parte a ser fabricado é projetado através de um sistema baseado em uma metodologia da *Webmachining* (sistema CAD/CAPP/CAM) onde depois que a parte é projetada é gerado automaticamente o plano de processo dela (PP) (Álvares, et al., 2005). Já através do sistema CAPP as ordens solicitadas pelo cliente são convertidas em ordens de trabalho as quais são utilizadas no plano mestre de escalonamento (*Master Scheduler*

Planning – MSP). O MSP e o PP servem como informação de entrada na Unidade de Gerenciamento.

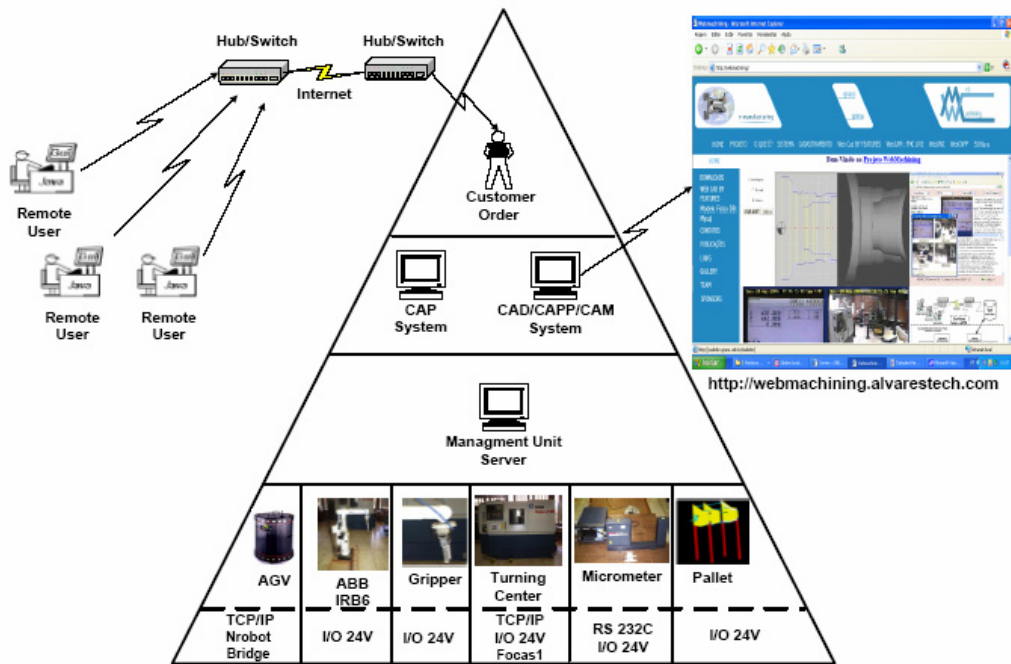



Figura 9.5 – Estrutura Hierárquica da FMC

A comunicação entre as diferentes estações de trabalho permite ter um controle remoto dos equipamentos da *FMC*, assim como também intercâmbio de dados entre eles durante o funcionamento da célula. A Figura 9.6 mostra a estrutura de comunicação da *FMC*. A tecnologia focas 1, desenvolvida pelo fabricante do torno CNC, permite realizar a comunicação entre a unidade de gerenciamento (*MgU*) e o centro de torneamento. O sistema radio *Ethernet* permite a comunicação com o robô móvel sem usar um cabo de *ethernet*. O usuário pode utilizar mecanismos ou serviços de redes tais como: ftp, *telnet, TCP/IP* (Nomadic Technologies, 1999).

A comunicação entre a unidade de gerenciamento (*MgU*) e a unidade de inspeção (micrômetro) é realizada através do uso do protocolo RS232. Já a comunicação entre a unidade de transporte e manuseamento de material (braço robô) e o Torno CNC é feita através do envio e recepção de sinais I/O 24V dos sistemas de PLC's.

Finalmente a comunicação entre o MgU e o centro de torneamento (Torno CNC) é feita através de o envio e recepção de sinais de 24V (CNC – MgU) e RS232 (MgU – CNC).

Uma interface de monitoramento desenvolvida permite a conversão dos sinais enviados para que possam ser lidos tanto pelo centro de torneamento como pelo MgU.

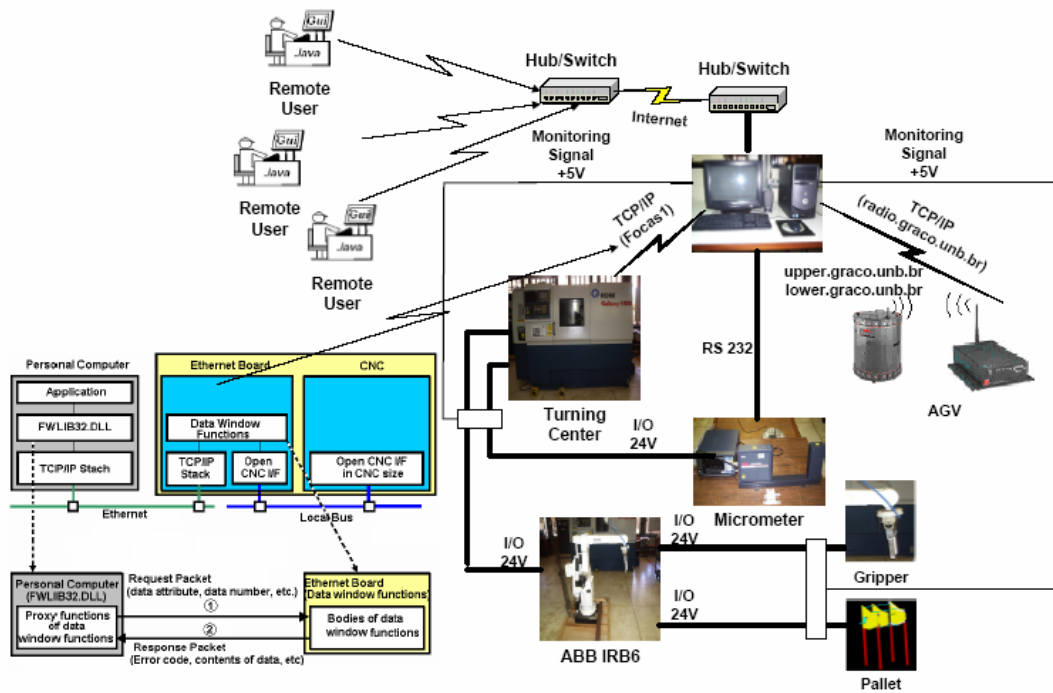



Figura 9.6 – Estrutura de comunicação da FMC

9.2.1.7 – Parte Física Implementada na FMC

Nesta seção serão descritos brevemente dois dos componentes da FMC que foram construídos para ser usados tanto no armazenagem de peças e de matéria prima como no posicionamento do manipulador de material (braço robô).

O armazém de peças, como se mostra na Figura 9.7 (a) foi construído em base a um modelo automatizado AS / RS para trabalhar em células flexíveis de manufatura. As dimensões do armazém consideradas na construção foram tomadas dependendo dos seguintes fatores:

a) comprimento e diâmetro das peças que serão fabricadas
b) abertura máxima da garra do manipulador (braço robô)
c) deslocamentos máximos do braço robô, permitindo a extração das barras de aço para serem usinadas.

a) b)

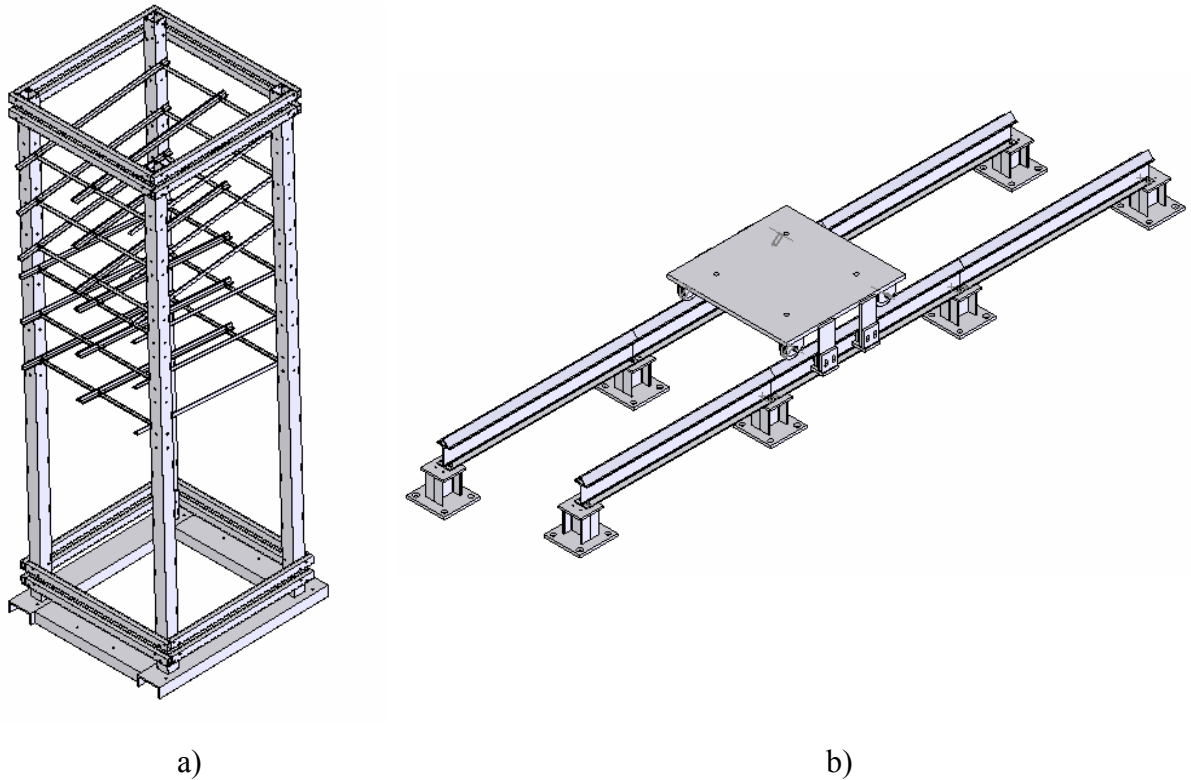



Figura. 9.7 – Componentes construídos da FMC. (a) Armazém de peças fabricadas e de matéria prima, (b) carro posicionador do manipulador de material – braço robô

O Apêndice D mostra as dimensões reais assim como também as vistas principais do armazém construído.

O carro para deslocamento do manipulador de material (braço robô), mostrado na Figura 9.7 (b) foi construído visando:

a) flexibilidade do braço robô nas tarefas de posicionamento tanto para a extração de material do armazém como posicionamento e extração de peças na placa do centro de torneamento, assim como também no posicionamento das peças para passar o controle de qualidade através do micrometro laser.
b) espaço livre durante a manutenção das máquinas principalmente do centro de torneamento, através do deslocamento do braço robô pelos trilhos ao extremo deles.

O Apêndice D mostra as dimensões reais assim como também as vistas principais do sistema de deslocamento do braço robô construído.

9.3 – MODELAGEM E IMPLEMENTAÇÃO DO AGV NA FMC – GRACO UnB

9.3.1 - INTRODUÇÃO

Neste capítulo serão descritas a modelagem e implementação do sistema de rastreamento de linhas e navegação utilizado pelo robô móvel, trabalhando como um AGV na FMC. Este sistema de navegação, baseado no sistema de visão do robô, é ideal para trabalhar em ambientes estruturados. O baixo custo de implementação é outro fator importante para se levar em conta, na implementação deste sistema.

Os *AGV's*, como já foi descrito no capítulo 4, são unidades de transporte que seguem rotas ou caminhos pré-especificados dentro da célula de manufatura. Comumente os caminhos são especificados por fios elétricos enterrados dentro do concreto do solo da fábrica. Os fios carregam os sinais elétricos AC detectadas por uma antena localizada no AGV. A escolha de múltiplos caminhos existentes pode ser feita dependendo das freqüências capturadas pelo veículo.

O sistema de rastreamento e navegação, baseado em visão, por ser simples, fácil de implementar, e ter um custo de operação relativamente baixo, motivam a pesquisa e garante procura de sistemas mais versáteis de nível industrial que satisfaçam otimamente as tarefas de ajuda durante as operações de manutenção. A flexibilidade, um termo muito importante na manufatura moderna, é outra razão para que este sistema tenha muita mais aceitação na industria que outros sistemas de navegação.

Uma técnica alternativa, também apresentada com mais detalhe na seção 5.1 e muito utilizada no campo da manufatura flexível é baseada na triangulação da localização do AGV através de faixas reflexivas montadas no teto ou paredes da fábrica. Um "*scanner*" localizado no veículo detecta a posição das faixas refletoras e vai proporcionando a posição onde o AGV está atuando em diferentes situações programadas. Varias técnicas adicionais e equipamentos para navegação baseados em sinais ou marcas ativas "*active beacons*" são descritas em (Borenstein, 1995).

No sistema de navegação que se está implementando, os fios elétricos são substituídos por uma linha pintada com um alto contraste em relação a cor do chão. A imagem desta linha é

capturada pela câmera localizada no robô móvel. O custo de pintar uma linha é baixo em comparação ao custo de implementação do sistema através de fios enterrados no chão, e muito menor que o sistema que usa laser para a localização do veículo.

9.3.2 – MODELAGEM DO AGV

9.3.2.1 – Modelagem do Ambiente de Atuação do AGV

O ambiente onde o robô móvel executará suas funções de navegação como AGV na FMC , realizando operações de transporte de material – "*material handling*", é determinado como um ambiente estruturado. Este ambiente possui como principais características a não variação da posição dos componentes ou equipamentos que conformam o dito ambiente e o não fluxo de pessoas durante o trabalho desenvolvido pelo AGV. Estas características são ideais nas industrias altamente automatizadas, já que a probabilidade de que aconteça algo imprevisto sempre é latente. É por este motivo que o robô móvel não está equipado com um só tipo de sensores; a fusão de vários deles faz com que a navegação seja feita com segurança.

A Figura 9.8 ilustra o ambiente estruturado onde o AGV realizará suas funções. Pode-se observar que tanto os equipamentos como os caminhos (que serão seguidos pelo AGV) são elementos pré-determinados com base em uma análise previamente desenvolvida, dependendo do processo de fabricação e a flexibilidade com que os trabalhos têm que ser feitos.

As características específicas mais importantes dos caminhos, como pode ser observado também na Figura 9.8, são: i) disposição estratégica de marcas que serão reconhecidas durante a navegação, usando sistema de visão do robô, e ii) alternativas nos caminhos para que o AGV consiga navegar por toda a célula, seja executando tarefas de transporte de material ou de supervisão da FMC.

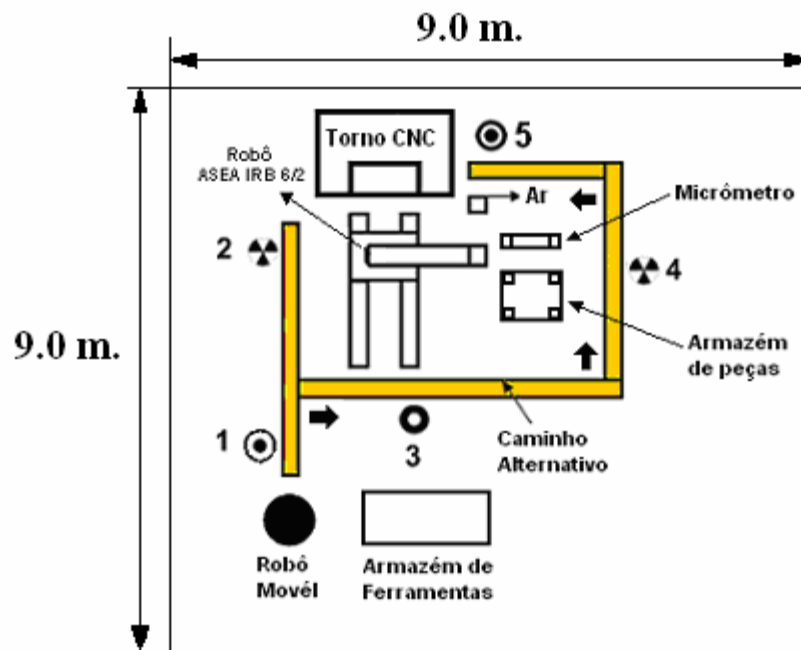



Figura 9.8 – Disposição das marcas e caminhos seguidos pelo AGV na FMC – Ambiente estruturado

9.3.2.2 – Classificação dos *AGV's* em Relação a seus Sistemas de Navegação

Os sistemas de controle e visão dos *AGV's*, dependendo do ambiente de atuação do veículo, podem ser classificados, de forma geral, em dois grandes campos: 1) num ambiente externo não necessariamente conhecido - "*Outdoor*" e 2) num ambiente interno fechado com características previamente conhecidas "*Indoor*"; a partir de esta primeira classificação geral destes sistemas pode-se derivar uma serie de subclassificações dos *AGV's*, baseadas em seus sistemas de controle, levando em conta que ditos ambientes, tanto externos como internos, podem se classificar também como estruturados e não estruturados. Uma vez definido o ambiente de atuação do AGV são apresentados os diferentes tipos de navegação existentes utilizadas em ditos ambientes. A Figura 9.9 apresenta esta classificação detalhada, destacando o tipo de navegação que será implementada no AGV na FMC. A cor amarela indica as características da classificação que se adapta ao sistema de navegação que se está implementando para ambientes externos com possibilidades grandes de ter obstáculos durante a navegação do robô e a cor verde indica as características de classificação para ambientes internos (livre de obstáculos), ambos estruturados.

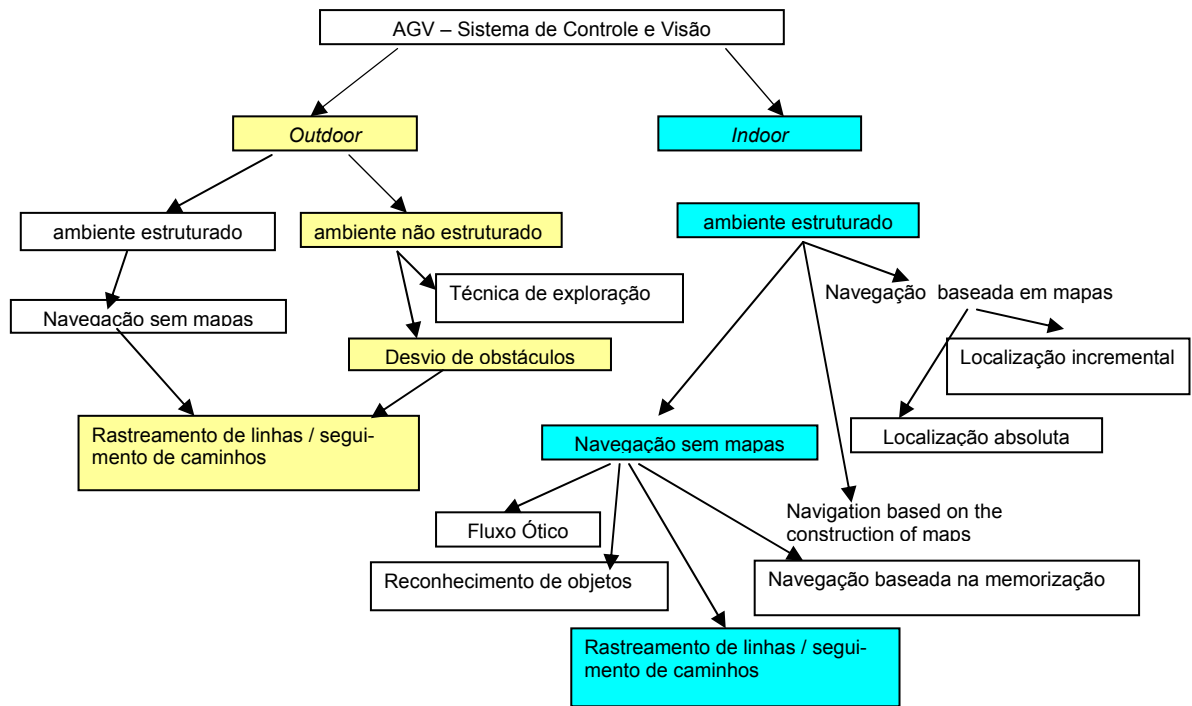



Figura 9.9 – Classificação dos *AGV's* (modificado – Ascione, 2003)

9.3.2.3 – Funções do AGV na FMC

Para o desenvolvimento da modelagem do sistema de funcionamento do AGV é necessário definir as funções especificas que realizara o robô móvel na FMC como unidade de transporte e manuseio de materiais – "*material handling*". Estas funções são: i) recepção das peças fabricadas (final, re-processo e "*default*"), colocadas pelo manipulador de material (braço robô) num sistema de armazenamento localizado na parte superior do robô móvel, formado por três espaços paralelos independentes para colocar cada tipo de peça, ii) transporte e recepção de ferramentas durante o "setup" ou manutenção das máquinas componentes da FMC e iii) supervisão do funcionamento da FMC através do sistema de visão do robô móvel (Booth, 1998).

As três funções assinaladas dão ao robô móvel a capacidade de trabalhar como um AGV na FMC. Estas funções podem ser executadas usando como sensor principal o sistema de visão do robô móvel tanto para o rastreamento dos caminhos desenhados no chão, assim como também para o reconhecimento das marcas ou sinais, os quais são baseados em símbolos geométricos ou alfanuméricos. O robô móvel se comunicará diretamente com o gerenciador da FMC, através do protocolo de comunicação TCP / IP, para a recepção e execução de suas tarefas como AGV.

9.3.2.4 – Modelagem do Sistema de Navegação do Robô Móvel

Como já foi mencionado na seção 9.2.2, o sistema de navegação que será utilizado pelo robô móvel é baseado no seu sistema de visão. Este sistema tem como sensor principal uma câmera *CCD* para captura de imagens no ambiente de navegação do robô.

Com base no sistema de navegação proposto para executar suas funções como *AGV*, o robô móvel navegará pela *FMC* seguindo caminhos desenhados no chão e reconhecendo marcas específicas (ver Fig. 9.8), as quais indicam uma ação que deverá ser executada por ele. Esta navegação, de forma resumida, é possível através do controle da inclinação das linhas usadas como caminhos seguidas pelo robô e pelo reconhecimento de marcas especificas utilizando inteligência artificial.

Diagramas UML

A modelagem do sistema de navegação do *AGV* será desenvolvida utilizando um *software* baseado em técnicas de objetos orientados através do uso da linguagem *UML* – "*Unified Modelling Language*". Este *software* permite a modelagem com os nove diagramas da *UML*: 1) Diagramas de Casos de Uso, 2) Diagramas de Classe, 3) Diagrama de Componentes, 4) Diagrama de Desenvolvimento, 5) Diagramas de Objetos, 6) Diagramas de Seqüência, 7) Diagramas de Colaboração, 8) Diagrama de Estados e 9) Diagrama de atividades. Nesta seção serão apresentados apenas os diagramas 1, 2 e 6 (Quatrani, 2000).

Diagrama de Caso de Uso

O diagrama de caso de uso, como mostrado na Figura 9.10, é um esquema gráfico que mostra algum ou todos os atores presentes no sistema que está sendo desenvolvido, assim como também seus comandos de utilização e interações no sistema (Cano, et al., 2005). Os comandos de utilização para o sistema são: i) envio da ordem para inicializar à navegação do robô móvel como *AGV*, ii) selecionar as tarefas que deverão ser executadas pelo *AGV* na FMC (durante o funcionamento da FMC ou durante a manutenção das máquinas), iii) receber informação do AGV quando a tarefa foi concluída, e iv) envio da ordem de finalização das funções de navegação do AGV ou de execução de alguma das tarefas que estão sendo desenvolvidas.

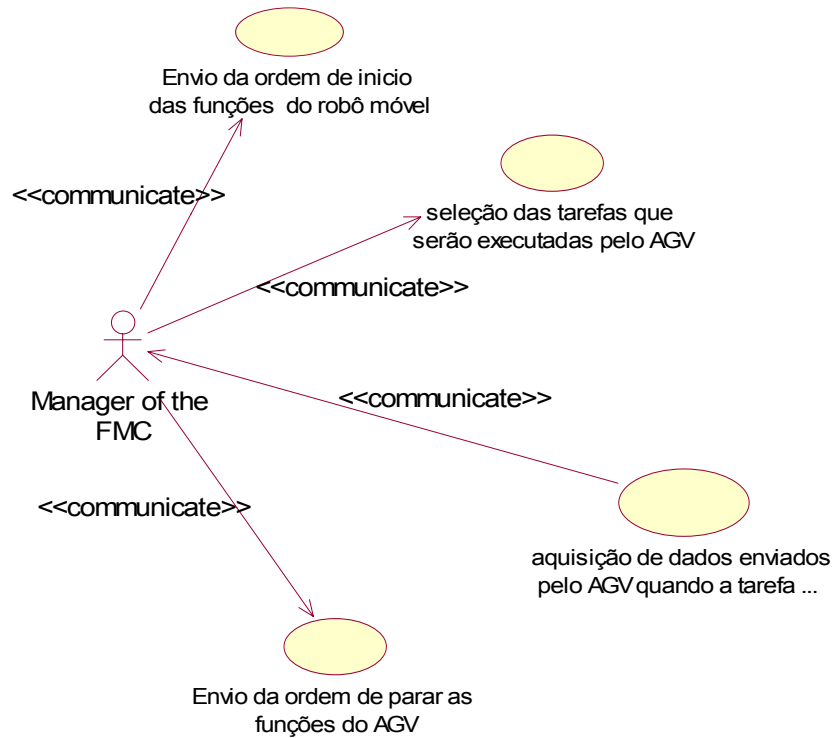



Figura 9.10 – Diagrama de caso de uso

Diagrama de Classe

Os diagramas de classe descrevem os tipos de objetos no sistema e os vários tipos de relacionamento estático que existem entre eles (Fowler, et al., 2000). Há dois tipos principais de relacionamento estático: Associação e Agrupação.

- associações: descrevem as relações entre duas classes. Por exemplo se o robô carrega peças dentro da máquina CNC, então a classe robô poderia associar-se com a classe máquina CNC, em termos de carregar partes. A Figura 9.11 ilustra este exemplo (Yang, et al., 2000).

Figura 9.11 – Associações entre classes

- subtipo : A Figura 9.12 mostra a superclasse sensor com suas subclasses: as classes sensor táctil, sensor de ultra-som, sensor infravermelho, sensor de visão e sensor de

posição. Estas subclasses não só herdam as características da superclasse sensor mas também têm seus próprios métodos e atributos.

- Agregação: descreve as relações "*a-part-off*" entre uma classe e seus componentes. Uma agregação é graficamente representada por um símbolo com forma de um diamante. A Figura 9.12 ilustra as relações entre a classe robô e seus componentes (classe atuadores e a classe sensor).

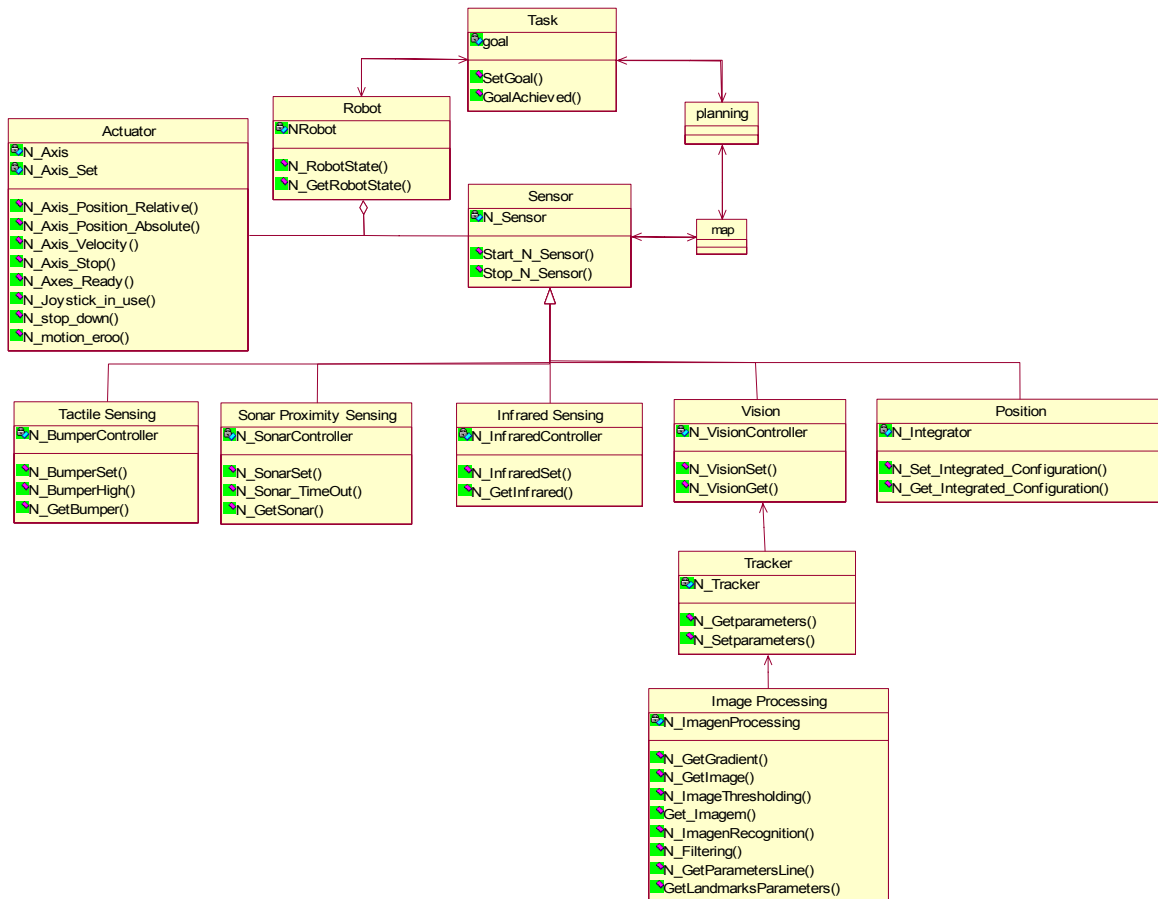



Figura 9.12 – Diagrama de Classes

O sistema de navegação, que usa como sensor principal o sistema de visão do robô móvel, trabalha também em conjunto com outro tipo de sensores (ultra-som, infravermelhos, táctil) os quais permitem evitar possíveis obstáculos imprevisíveis durante o funcionamento da FMC. As diferentes classes definidas para a modelagem da navegação do robô são: classe tarefa, classe robô, classe sensor, classe acionador, classe sensor táctil, classe sensor ultra-som, classe sensor infravermelho, classe posição, classe "*tracking*", classe processamento de imagens, classe visão, classe planejamento de rotas, classe mapas.

O Apêndice C mostra com detalhes cada uma das classes com seus respectivos atributos e métodos.

Diagrama de Seqüências

Os diagramas de seqüência, também chamados de diagramas de interação (Fowler, 2000), são modelos que descrevem como grupos de objetos colaboram em algum comportamento. Tipicamente, um diagrama de interação captura o comportamento de um único caso de uso. O diagrama mostra vários objetos e as mensagens que são passadas entre estes objetos em um caso de uso.

O diagrama de seqüência apresentado na Figura 9.13, através da interação dos objetos usando os métodos respectivos, mostra a seqüência de passos realizados no sistema de navegação do robô para cumprir suas funções como AGV na FMC. Esta seqüência começa com a seleção da tarefa, que o robô deverá executar. Esta seleção é feita pelo gerenciador da célula responsável por transmitir esta informação à classe *Tarefa* ("*Task Class*"). Esta classe por sua vez é a responsável de transmitir esta informação à classe *Robô* ("*Robot class*") como o objetivo que deverá alcançar o robô durante sua navegação ("*goal*"). A classe robô fazendo uso das classes acionador e sensor ("*Actuator class*" e "*Sensor class*") executa a seqüência de passos para realizar a navegação do robô. A classe robô depois de receber a ordem de começo das funções de navegação inicia os sensores do robô através da classe *Sensor* ("*class Sensor*") a qual inicia as funções de cada um dos sensores existentes no robô (infravermelhos, ultra-som, táctil, visão e posição) através do uso de suas subclasses. Uma vez capturadas as imagens através do sensor de visão (imagem RGB) a classe *Processamento de Imagens* (*"Image Processing Class"*) realiza as operações de pré-processamento (transformação da imagem RGB a cores de cinza, filtragem, transformação de perspectiva, binarização, etc) e computa os dados de orientação e posicionamento do robô móvel assim como também faz o reconhecimento das marcas dispostas na FMC. Seguidamente a imagem pré-processada e passada para ser analisada pela classe "*tracking*" através da comparação dos dados obtidos com os erros mínimos permitidos, realizando uma retro-alimentação, recalculando constantemente estes dados e comparando-os com os obtidos depois do novo posicionamento do robô. Finalmente, o robô atinge seu objetivo quando uma marca é reconhecida a qual indicará uma ação que deverá tomar (parar, virar, virar e voltar, etc).

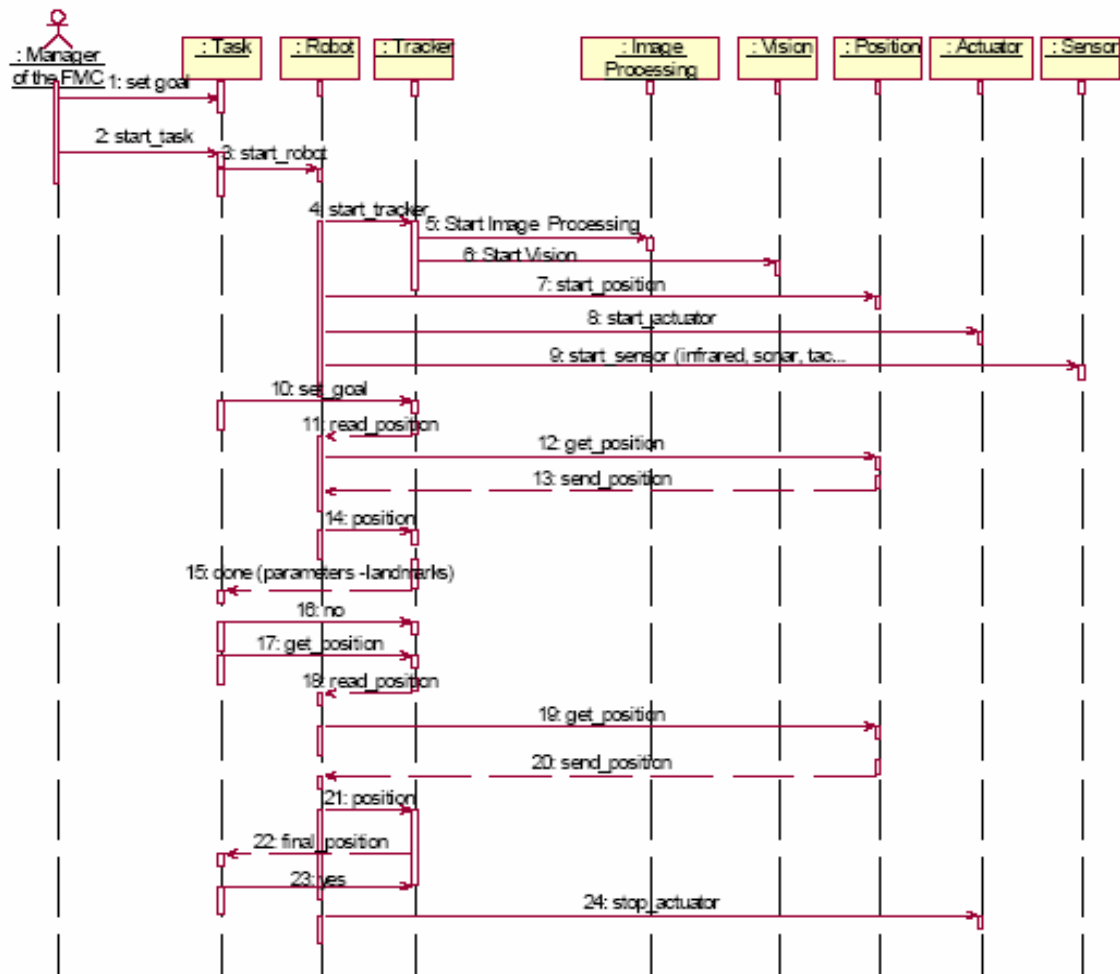



Figura 9.13 – Diagrama de Seqüências

9.3.2.5 - Detalhe das funções do robô móvel durante o funcionamento da FMC

O gerenciador da FMC é o responsável de enviar as ordenes das tarefas que deverão ser executadas pelo AGV. Estas ordens são basicamente relacionadas ao posicionamento e localização do AGV no ambiente da FMC, definidos pelas marcas desenhadas no chão como ilustrado na Figura 9.8.

Uma vez que se deu inicio ao funcionamento da FMC o robô móvel se posicionará no ponto (1), ilustrado na Figura 9.8, o qual indica o lugar de partida para realizar suas funções como AGV. Seguidamente o robô móvel receberá outra ordem, proveniente do gerenciador da FMC, para se posicionar no ponto (2) a espera que o manipulador de material (braço robô) coloque a peça fabricada, previamente qualificada pelo sistema de controle de qualidade (micrômetro laser), no espaço correspondente do armazém localizado na parte superior do robô móvel. Este processo de posicionamento do robô móvel entre os pontos (1) e (2)

é repetitivo durante o funcionamento da FMC até que receba a ordem de parar as funções como AGV e se posicionar no ponto (1) a espera de outra ordem.

9.3.2.6 - Detalhe das funções do robô móvel durante o "*set-up*" ou manutenção das máquinas da FMC

Quando a FMC não está em funcionamento, deixando de lado a possibilidade de falta de trabalho, o robô móvel estará executando tarefas de transporte de ferramentas desde o armazém ao lugar onde fica a máquina em "*set-up*" ou manutenção. O caminho alternativo (Figura 9.8) é desenhado para a realização destas funções, os pontos (3), (4) e (5) são habilitados dependendo se máquina está em "*set-up*" ou manutenção; e assim o robô móvel poda se posicionar no lugar adequado.

9.4 – SET-UP DO ROBÔ MÓVEL

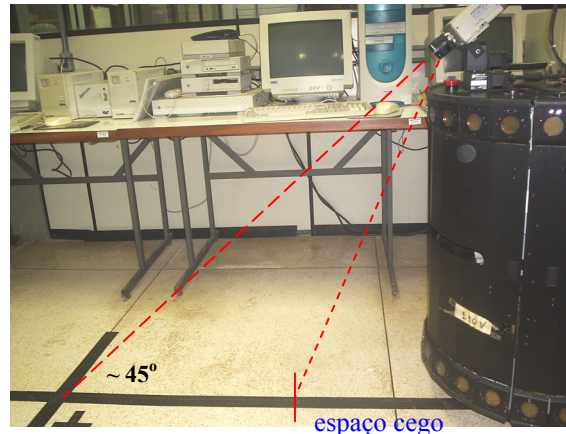
O robô móvel que será integrado na FMC para trabalhar como AGV, é o Nomad XR4000 apresentado na Figura 9.14. Nesta figura podem ser também apreciados alguns dos componentes sensoriais que formam parte do *hardware d*o robô (sensores de ultra-som, visão e infravermelho). O *hardware* principal de controle do robô é formado por dois computadores, "*upper*" e "*lower*" rodando no sistema operacional Linux. Na parte superior do robô está montada a unidade de direção dos movimentos da câmera CCD "*Pan-Tilt Unit*" (PTU) e captura das imagens e realizada através da placa de captura de imagens, a qual permite taxas de atualização de até 30 fps (imagens por segundo) e resoluções de 640 x 480, e uma câmera de vídeo a cor. Mais detalhes do *hardware* do robô móvel foram apresentados na seção 8.1.1.

(a) (b)

Figura 9.14 – Robô móvel. a) Robô móvel Nomad XR4000, b) Posicionamento da câmera a 45º aproximadamente.

O sistema de visão do robô móvel trabalha com os seguintes parâmetros de configuração: 30 fps e em resolução baixa de 128 x 128.

Como pode ser visto na Figura 9.14 (b) a localização da câmera não é apropriada para conseguir a detecção das linhas. A ótima localização da câmera deveria ser perto do chão, posicionada na parte inferior do robô. A máxima inclinação, para abaixo, da orientação do PTU é aproximadamente 45º quando a câmera do robô esta capturando as imagens da linha. Esta disposição da câmera leva a ter um "espaço cego" na frente do robô de aproximadamente 370mm. a qual deve ser controlada pelo sistema de navegação do robô através dos diferentes tipos de sensores.

9.5 – PROCESSAMENTO DE IMAGENS

O processamento de imagens utilizado para a navegação do AGV envolve operações de baixo nível como, realce, detecção e computação da orientação de linhas, identificação de interseções entre linhas que definem os diferentes caminhos e detecção e reconhecimento de sinais.

9.5.1 – Pré-Processamento de Imagens

O pré-processamento das imagens é realizado basicamente em as seguintes etapas: a) filtragem, b) transformação de cores, e c) limiarização por histograma.

O ruído é uma variável que sempre estará presente na aquisição das imagens. À medida que o ruído é incrementado ou diminuído os filtros correspondentes são aplicados para depois binarizar a imagem através do uso de um valor limiar "*threshold*".

A primeira etapa de filtragem, é feita utilizando um filtro passa baixa. A filtragem passa baixa da imagem é feita para reduzir o nível de ruído, geralmente presente em sistemas que trabalham com sensores. A aplicação desta filtragem e realizada através do cálculo do valor médio de quatro pixéis vizinhos, $1/4((i,j)+(i,j+1)+(i+1,j)+(i+1,j+1))$, utilizando o valor dado como entrada para o novo pixel na imagem reduzida. A redução dos pontos da imagem a um quarto do valor inicial garante um menor custo computacional de processamento posterior da imagem. Na seção 7.4.2 foram apresentados com mais detalhes o funcionamento deste tipo filtro.

O seguinte passo a ser realizado é a transformação de cores. A placa de captura de imagens tem como padrão a captura das imagens em três componentes RGB (red, green, blue) separadamente, embora os métodos de processamento de imagens sejam definidos em geral para imagens monocromáticas. A transformação de cores de RGB para o padrão HSV(hue, saturation, value) permite que a imagem resultante seja monocromática, que haja uma grande redução do espaço de armazenamento e que se reduza o tempo de processamento nas operações posteriores. Assim, com base nos três componentes da imagem original, gera-se uma imagem monocromática com o valor médio das imagens originais, dada pela componente V do padrão HSV (Tourino, 2000). A equação 7.1 mostra o cálculo da componente V.

$$V = \frac{R+G+B}{3} \quad (9.1)$$

O efeito desta transformação é a redução de informação originalmente disponível nas três componentes RGB. Esta redução de informação é refletida na perda de informações associadas às cores da cena.

Depois de realizar a captura, filtragem e transformação de cores pode-se realizar a binarização da imagem. Deve-se notar que a metodologia de binarização utilizada é aplicada diretamente à imagem original previamente processada através das etapas descritas anteriormente. O objetivo desta etapa é conseguir a segmentação das linhas em relação ao seu fundo. A binarização da imagem é obtida da aplicação da equação 7.20 através do uso de um valor limiar ou "*threshold*" definido. O valor limiar definido para o processo baseia-se no histograma da imagem original equalizada (ver seção 7.4.2). O algoritmo utilizado proporciona o valor de intensidade com maior número de pixeis contidos na imagem equalizada, o qual será utilizado como valor limiar.

Outra metodologia avaliada é apresentada por Beccari (1997) para a obtenção do valor limiar ótimo é baseado na Equação (9.2):

$$T = T_o - kx_m \tag{9.2}$$

onde T_o é a média dos valores de intensidade dos pixéis, k é o coeficiente de ajuste, e x_m é a média local dos valores de intensidade sobre uma vizinhança de pixéis.

O pré-processamento das imagens deve ter um custo computacional baixo o que implica rapidez na execução dos algoritmos.

Na Figura 9.15 (a) mostra a figura original capturada pelo robô móvel. A Figura 9.15 (b) mostra a imagem pré-processada (filtragem, transformação de cores e binarização).

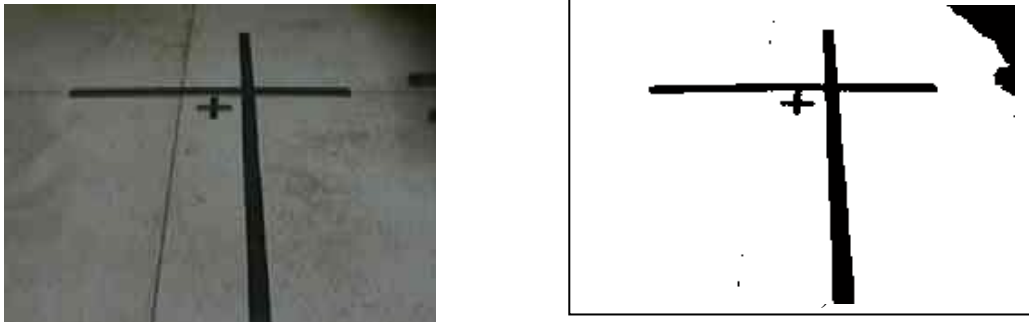



Figura 9.15 – a) Imagem original capturada pela câmera do robô móvel, b) Imagem pré-processada para detecção de linhas e sinais.

9.5.2 – Detecção de Linhas

A operação visual principal necessária para a navegação do AGV é calcular a orientação do caminho desenhado no chão. Esta operação estima a inclinação ou pendente do caminho trabalhando-o como uma linha através do uso de um par de pontos da imagem.

O caminho que o AGV estará seguindo não necessariamente é uma linha reta, como pode ser visto na Figura 9.16. O AGV têm outras alternativas de movimentação para poder navegar pela FMC, necessárias para cumprir com suas funções de transportador de material. Assim, para evitar as mudanças bruscas de direção nos caminhos seguidos, a imagem capturada é analisada através de duas faixas na parte inferior da imagem como apresentado na Figura 9.16.

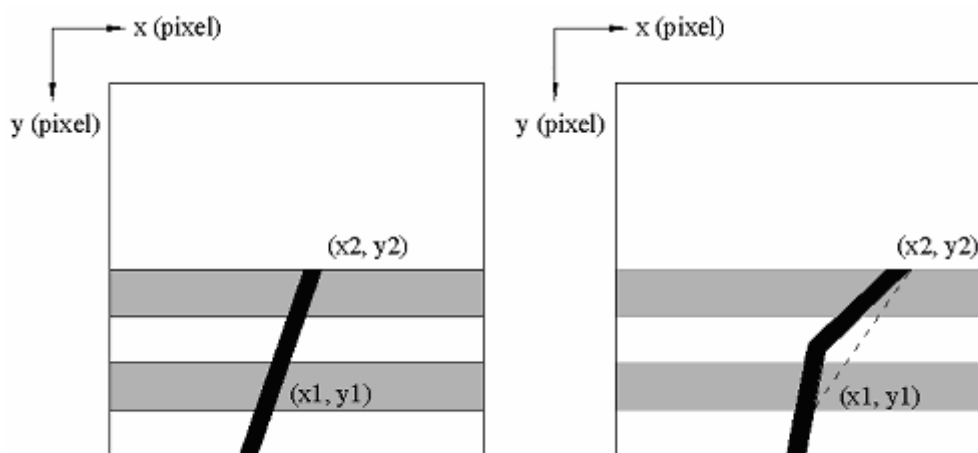



Figura 9.16 – Pontos pré-selecionados para o cálculo da direção dos caminhos

Nesta parte do trabalho serão analisados dois dos métodos mais comumente utilizados para o reconhecimento de caminhos rastreados através de um sistema de visão: 1) extração de pontos centrais das linhas em determinadas faixas da imagem binarizada e 2) reconhecimento de linhas através do algoritmo da transformada de Hough "*Hough transform*". O segundo método é apresentado como alternativa comumente utilizada no reconhecimento de linhas em imagens, mas não será utilizado no desenvolvimento deste trabalho.

O primeiro dos métodos é baseado na extração de dois pontos, $(x1, y1)\, e\, (x2, y2)$, de duas faixas da imagem como mostrado na Figura 9.16. Dependendo da luminosidade do ambiente, as faixas escolhidas na imagem, para tirar os pontos, pode variar. A partir do vetor, representados por zeros ou uns, das faixas escolhidas da imagem binarizada é calculado o centro da pista, para cada uma das faixas, através de um varrido delas, para posteriormente, utilizando a fórmula de média ponderada como se mostra na equação 9.3 se possa calcular a localização do pixel central (Costa, 2003):

$$Cx = \frac{\sum_{i=0}^{i=m} x_i . i}{\sum_{i=0}^{i=m} x_i} \qquad 9.3$$

onde:

x_i é o vetor binarizado na posição i ;

m é a largura da imagem;

C_x é o centro da pista para cada uma das linhas.

No caso da pista não ser encontrada (por esta ter terminado), o sistema atribui ao valor do centro da pista a coluna zero. Este valor é utilizado para avisar ao módulo de controle que pare o movimento do robô. Já com os dois pontos obtidos e depois de aplicar a transformação de perspectiva; a equação (9.4), referente à equação da reta, pode calcular a orientação da reta. Esta orientação posteriormente será comparada com o erro mínimo de orientação para executar o controle do robô

O segundo método apresentado para a detecção de caminhos é baseado na transformada de Hough. Este método é usado para detectar linhas retas na imagem capturada e assim poder definir a direção do caminho. A idéia da transformada de Hough é aplicar na imagem uma

transformação tal que todos os pontos pertencentes a uma mesma curva sejam mapeados num único ponto de um novo espaço de parametrização da curva procurada (Gonzalez, 2000). A Figura 9.17 apresenta um exemplo da imagem capturada, pré-processada, com duas das linhas detectadas de interesse para conhecer a direção do caminho que está sendo seguido pelo AGV. Uma vez achadas estas duas linhas na imagem são tirados pelo menos dois pontos arbitrários destas linhas para poder calcular sua posição real, e assim conhecer a inclinação real do caminho seguido pelo AGV, e corrigir o erro de acordo com os mínimos de inclinação permitidos. O pré-processamento realizado na imagem capturada, além da filtragem de ruído e transformação de cores, é a aplicação do filtro Sobel ou Roberts para a detecção das bordas da imagem.

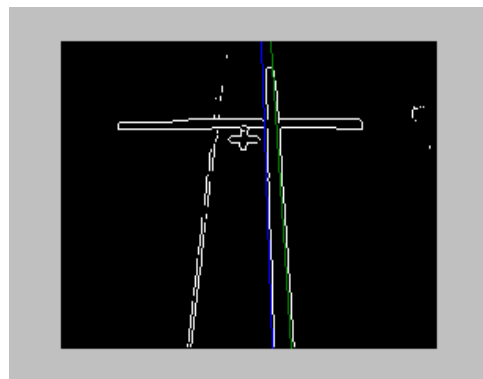



Figura 9.17 – Imagem pré-processada (Filtro Sobel) e Aplicação da transformada de Hough para a detecção de linhas.

9.5.3 - Remoção da distorção de perspectiva

As coordenadas dos pontos identificados através do uso de qualquer dos dois métodos são afetados pela deformação da perspectiva. O mapeamento inverso de perspectiva (IPM), explicado com detalhe na seção 7.4.4, é aplicado para corrigir a direção do caminho afetada por esta distorção tal como se pode apreciar na Figura 9.18.

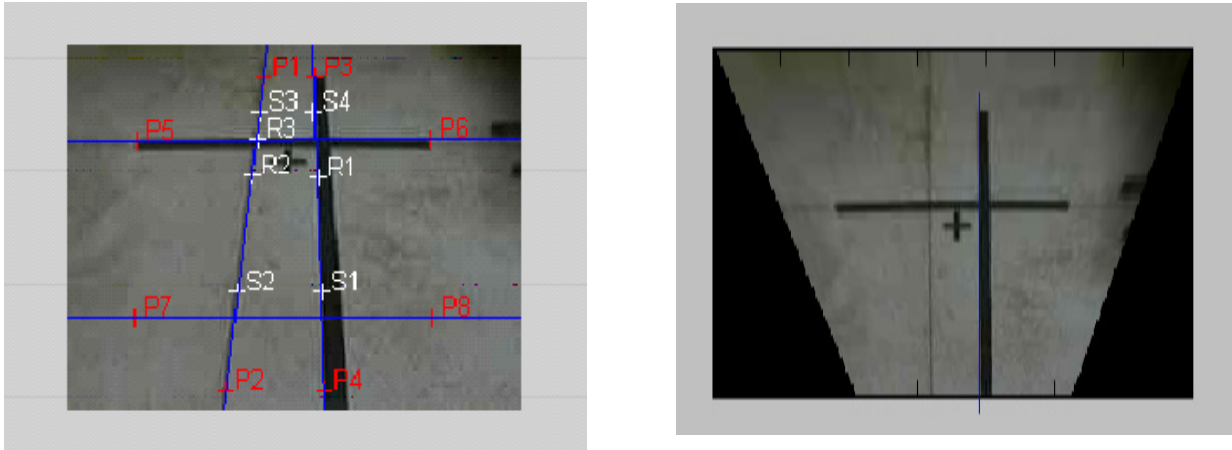



Figura 9.18 – Remoção da distorção de perspectiva: imagem original (esquerda) ; imagem com a perspectiva corrigida (direita).

Com o objetivo de reduzir o custo computacional, refletido no tempo de processamento utilizado (*ms*), a imagem total não precisa ser transformada; só dois pontos são necessários ser mapeados para ajustar a inclinação do caminho seguido pelo AGV.

9.5.4 – Reconhecimento de Sinais

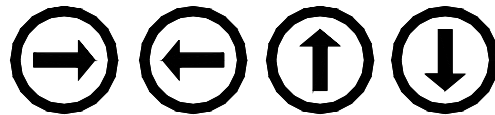
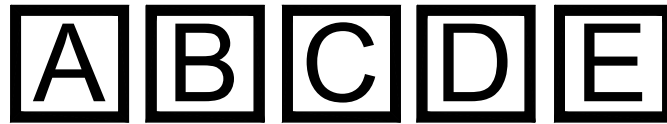
Outra das tarefas que serão realizadas pelo AGV, além da detecção de linhas nos caminhos, é a identificação e reconhecimento de sinais ou marcas, as quais sugerem ou definem o comportamento e navegação do AGV. Estes sinais, ou marcas, são utilizados durante a navegação do robô móvel e cumprem basicamente com a função geral de indicar o começo e fim dos caminhos, assim como indicar os pontos de localização do robô onde realizará algumas das tarefas assinaladas pelo gerenciador da FMC. A forma externa destes sinais é representada por símbolos simples (círculos, quadrados, triângulos) e a forma da parte interna por (números, letras, setas). A forma externa define o tipo de sinal, indicando se elas são sinais de trânsito ou de localização. A Figura 9.19 mostra os diferentes símbolos, recomendados por Beccari (1997), utilizados durante a navegação do AGV.

b)

Figura 9.19 - Símbolos de trânsito e alfanuméricos, (a) Conjunto de símbolos de trânsito (b) Conjunto de símbolos de localização no ambiente (modificado – Beccari, 1997).

Segundo Beccari (1997), os sinais simbólicos são preferencialmente utilizados, em vez de sinais baseados em códigos de barras ou geométricos, em ambientes onde há potencial presença de pessoas ou outros sistemas de transporte de carga, e assim possam ser reconhecidos evitando prováveis acidentes.

Atualmente na industria, principalmente nas áreas de manufatura e armazenamento, há uma grande variedade de sinais ou marcas comumente usadas, tanto em sistema manuais como autônomos. A Figura 9.20 mostra algumas das marcas mais freqüentemente utilizadas (Amat, et al., 2001).

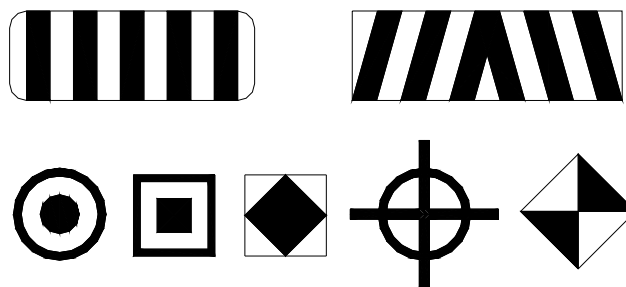



Figura 9.20 – Sinais "*landmarks*" comumente utilizadas na industria para a navegação dos *AGV's* (modificado – Amat, 2001)

Amat (2001), partindo dos símbolos mostrados na Figura 9.20, selecionou um conjunto destes com uma potencial característica em comum: maior precisão em sua localização e conseqüentemente um melhor posicionamento do AGV pode ser obtido. A Figura 9.21 mostra os quatro padrões testados (cores originais e cores invertidas). Todos estes padrões têm em comum o alto grau de contraste com o fundo e sua eleição é fundamentada na pouca influência do ruído ocasionado pelo processo de segmentação devido aos elementos visíveis do fundo. Segundo este autor os padrões que são mais susceptíveis ao ruído, ocasionando maior erro de posição, são os dois primeiros, já os dois seguintes, devido a sua geometria radial, mostram ser quase insensíveis ao ruído ocasionado pelo fundo.

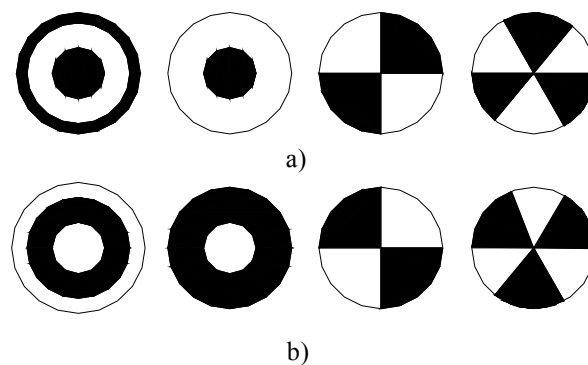



Figura 9.21 – a) marcas padrões "landmarks" testados b) marcas com as cores invertidas

Estes padrões, pelo fato de trabalharem através do uso de um sistema de visão, estão predispostos aos efeitos de distorção ocasionados pela transformação de perspectiva da imagem capturada pela câmera (ângulo formado pelo eixo óptico da câmera e o campo de visão) e a não absoluta horizontalidade do posicionamento do AGV. Todos estes erros são analisados por Amat (2001) concluindo que as marcas mostradas na Figura 9.21 são as idôneas para serem usadas em uma navegação com um sistema de visão como sensor principal.

Uma vez definidas as marcas que serão utilizadas, e o significado que elas terão durante a navegação do AGV, o passo seguinte é identificá-las na imagem capturada. Tanto a identificação e a classificação das marcas são realizadas através do uso de técnicas baseadas em redes neurais. A imagem binarizada é examinada através de um algoritmo que funciona como uma "janela" de identificação das regiões com potencial presença de marcas na imagem (Adorni, 1996). Depois que as regiões foram identificadas o algoritmo faz o reco-

nhecimento das marcas com formas compatíveis ao conjunto de marcas pré-selecionadas como padrões (Adorni, et al., 1997).

A percepção da forma é executada através de um conjunto de neurônios auto-associados, cada um relacionado a uma das diferentes formas armazenadas como padrões e capaz de reproduzir como saída o mesmo padrão se e somente se é um único padrão conhecido pela rede neural que está sendo analisado num instante durante a navegação do veículo. O reconhecimento do símbolo, então, é realizado através do uso de uma rede neural atuando só na parte selecionada pelo algoritmo identificador da marca ou símbolo na imagem capturada. A dimensão da imagem capturada é de 128 x 128, já a dimensão da parte selecionada da imagem pertencente ao símbolo tem um tamanho de 16 x 16 pixéis.

O modelo de rede neural utilizado é MLP – *"multilayer perceptrons"* ou Perceptrom Multicamadas (ver seção 7.4.6.3) treinada com o algoritmo *"backpropagation"* utilizando 10 ou mais formas diferentes para cada uma das marcas ou símbolos apresentados nas Figuras 9.20 e 9.21. O propósito de treinar a rede neural com uma grande variedade de formas diferentes, é reconhecer os símbolos padrões ainda quando estes estejam afetados pela transformação de perspectiva e evitar o uso do algoritmo IPM o qual aumenta a carga computacional (Autio, et al., 2001).

Uma vez que as marcas são reconhecidas, novas informações são enviadas ao sistema de movimentação do robô (X, Y, θ) para sua nova localização através do uso da estrutura $struct\, N_Integrator$, modificando os ditos parâmetros através do uso de $N_SetIntegratedConfiguration$. Variações nos parâmetros de velocidade e aceleração podem ser também incluídos.

9.6 - CONTROLE DA ORIENTAÇÃO E POSICIONAMENTO DO ROBÔ

9.6.1 – Controle da Orientação do Robô

O controle da orientação e direção do robô (*"steering"*) usando características das imagens capturadas, através de um sistema de visão, tem sido tema de pesquisa principalmente no campo de automação de veículos de estrada (Thorpe, et al., 1988), (Bertozzi, et al., 2000) e

(Collado, et al., 2003). Entre as diferentes aplicações utilizadas, baseadas em algoritmos de visão desenvolvidos, figuram técnicas de controle de direção usando pontos da imagem, com transformação de perspectiva, para o rastreamento de linhas dos caminhos seguidos pelo veículo.

O controle da direção do AGV, durante o rastreamento da linha é relacionado diretamente com o controle do erro de orientação em relação ao mínimo permitido. Este erro de orientação ao mesmo tempo é relacionado com o ângulo α'_t mostrado na Figura 9.22 o qual é calculado através da equação (9.4) como o arco tangente da pendente m .

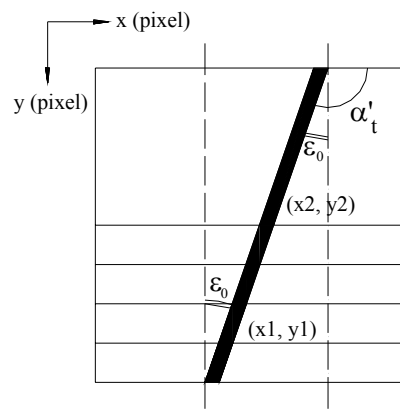



Figura 9.22 – Esquema dos ângulos que devem ser controlados durante o rastreamento de linhas.

$$m = \frac{y_1 - y_2}{x_1 - x_2} \quad \rightarrow \quad \alpha'_t = \arctan(m) \qquad (9.4)$$

Já o erro de orientação propriamente dito ε_o e calculado através da equação (9.5) o qual é mostrado também na Figura 9.22.

$$\varepsilon_0 = 90 - \alpha'_t \qquad (9.5)$$

Este erro pode se negativo ou positivo dependendo se a linha rastreada está inclinada à direita ou à esquerda, respectivamente.

Como a inclinação da linha rastreada pode estar variando entre [-90º, 90º], é necessário analisar os casos onde a pendente m, pode tomar valores positivos ou negativos, depen-

dendo da inclinação da linha. A equação (9.6) mostra um analise para estas variações de m (Borenstein, 1998).

$$\begin{aligned} m > 0 \;&\Rightarrow\; \arctan(m) = \alpha_t > 0 \;\Rightarrow\; \varepsilon_0 = 90 - \alpha_t \\ m < 0 \;&\Rightarrow\; \arctan(m) = \alpha_t < 0 \;\Rightarrow\; \varepsilon_0 = -(90 - \alpha_t) \end{aligned} \tag{9.6}$$

A Figura 9.23 ilustra as variações na inclinação da linha rastreada através da variação do ângulo α_t, assim como também os diferentes erros (ε_0) que se apresentam partindo destas variações.

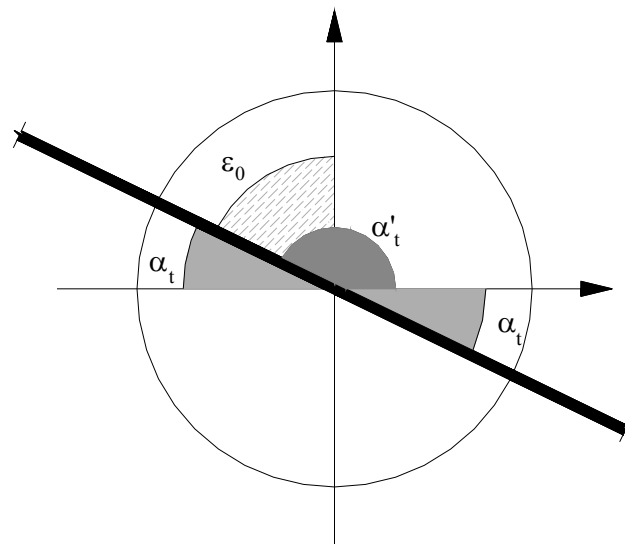



Figura 9.23 – Análise das inclinações da linha rastreada

9.6.2 – Controle do Posicionamento do Robô

O controle do posicionamento do robô móvel pode ser calculado pela seguinte equação:

$$\varepsilon_p = l.\sin(\varepsilon_o) \qquad 9.7)$$

onde l é soma do "espaço cego" do robô (d) e a distancia (y_c) até a interseção da linha na direção do robô e a linha que passa pelo caminho. A Figura 9.24 mostra a geometria que relaciona as distancias para o cálculo do erro de posicionamento.

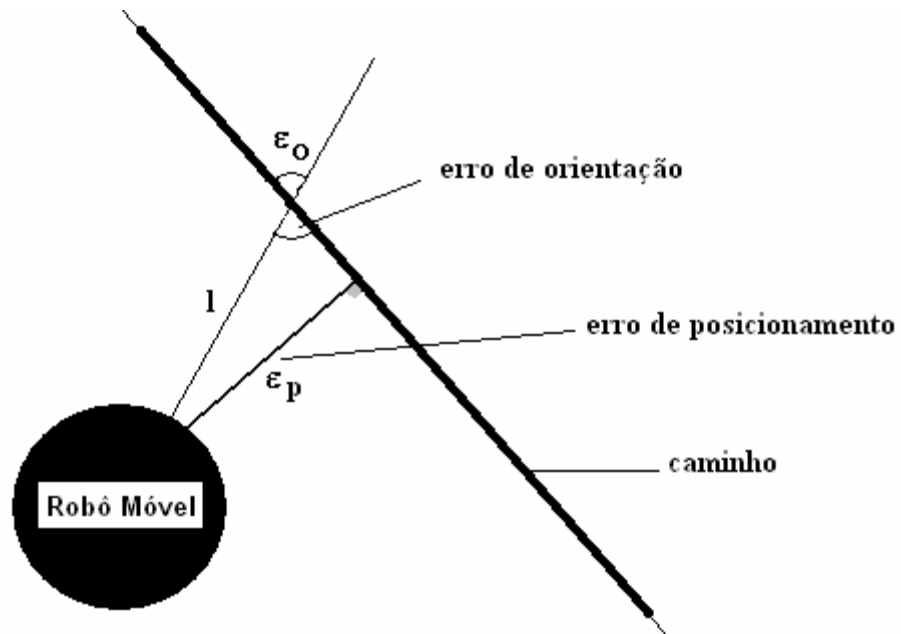



Figura 9.24 - Geometria para o cálculo do erro de posicionamento

10 - RESULTADOS EXPERIMENTAIS

O sistema de navegação do robô móvel trabalha seguindo uma seqüência de algoritmos, baseados em visão computacional segundo as metodologias descritas nos capítulos anteriores, sendo os resultados obtidos nas diferentes etapas, apresentados de forma resumida ao longo deste capitulo.

10.1 - RESULTADOS OBTIDOS NAS ETAPAS DE AQUISIÇÃO DE IMAGENS E PRÉ-PROCESSAMENTO

O processo de aquisição de imagens e pré-processamento, conforme discutido anteriormente, compreende etapas de filtragem, transformação de cores, e limiarização por histograma.

Uma vez que a imagem é adquirida, através da câmera em formato RGB, a filtragem é realizada utilizando um filtro passa-baixa de dimensão (2 x 2) feita para reduzir o nível de ruído. A fim de otimizar as operações de processamento utilizou-se um método que combina a redução de resolução simultaneamente com a filtragem passa-baixa (filtro média), através do cálculo do valor médio de quatro pixéis vizinhos e utilizando esse valor como entrada para o novo pixel na imagem reduzida (ver Seção 7.4.2 e Seção 9.5.1).

O passo seguinte é a transformação de cores necessário para converter a imagem com formato RGB (*red, green, blue*), adquirida pela placa de captura de imagens Matrox Meteor, para o padrão HSV(*hue, saturation, value*). Com base nos três componentes da imagem original gera-se uma imagem monocromática com o valor médio das imagens originais dada pela componente V (Seção 9.5).

O passo seguinte e a limiarização da imagem ou binarização. Como já foi abordado na seção 9.5, o objetivo desta etapa é conseguir a segmentação das linhas em relação ao seu fundo. A binarização da imagem é obtida da aplicação da equação 7.20 através do uso de um valor limiar ou "*threshold*" definido. O valor limiar definido para o processo baseia-se no histograma da imagem original equalizada (ver seção 7.4.2). O algoritmo utilizado proporciona o valor de intensidade com maior número de pixéis contidos na imagem equalizada, o qual será utilizado como valor limiar.

Os passos acima descritos são aplicados a todas as imagens processadas pelo sistema. As Figuras 10.1, 10.2 e 10.3 mostram as diferentes imagens (filtradas, com transformação de cores, e binarizadas) para os três símbolos que posteriormente deverão ser reconhecidos na imagem. Pode-se notar que para os três tipos de imagens à imagem com valor limiar ("threshold") igual a 30 é a que melhor separa a imagem de interesse do fundo (limiarização por histograma). Este valor foi fixado para o nível de luminosidade do ambiente de trabalho onde são realizados os testes.

Estes algoritmos de pré-processamento são executados dentro do programa principal "*image_get_exe.c"* rodando na máquina "*upper*" do robô móvel num ambiente "*linux*".

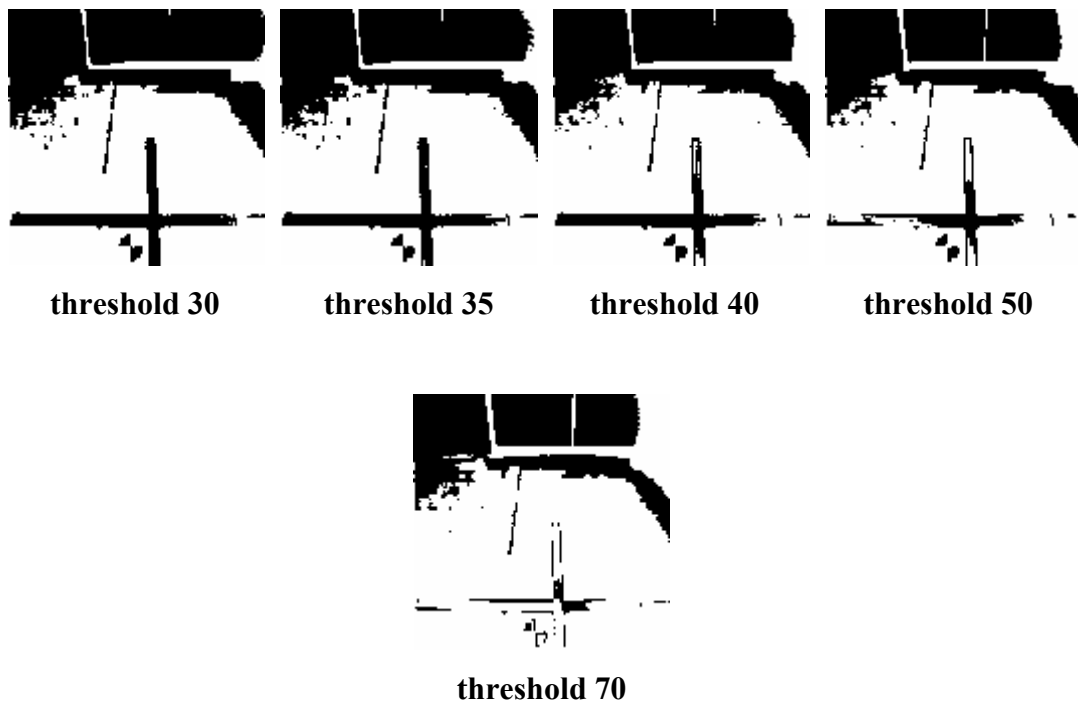



Figura 10.1 – Sequencia de imagens pre-processadas – símbolo 1

Figura 10.2 – Sequencia de imagens pre-processadas – símbolo 2

Figura 10.3 – Sequencia de imagens pre-processadas – símbolo 3

10.2 - RESULTADOS OBTIDOS NAS ETAPAS DE REMOÇÃO DA DISTORÇÃO DE PERSPECTIVA

A transformação de perspectiva, explicado com detalhe na seção 7.4.4, é aplicada para corrigir a direção do caminho afetada por esta distorção a qual é provocada pela inclinação da câmera (45º) para realizar o rastreamento dos caminhos e sinais desenhados no chão. A Figura 10.4 mostra as diferentes imagens afetadas pela distorção de perspectiva para um desvio na orientação do robô de 11º. O programa "*ipm.m*" desenvolvido no software "*matlab*" remove estas distorções apresentando as imagens como uma vista desde o céu.

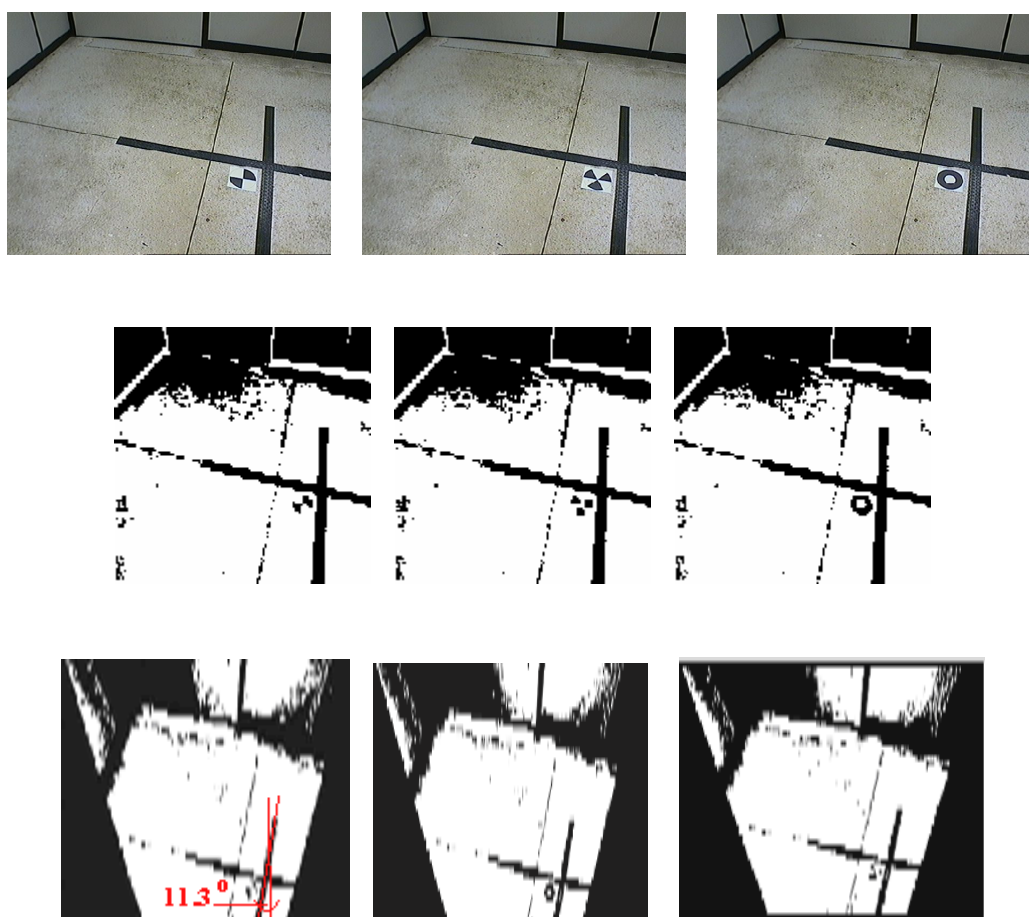



Figura 10.4 – Imagens afetadas pela distorção de perspectiva para um desvio na orientção do robô de 11º

A Figura 10.5 apresenta os parâmetros, intrínsecos e extrínsecos, de configuração da câmera do robô móvel utilizados na aplicação das fórmulas de remoção da distorção de perspectiva das imagens capturadas pela câmera do robô.

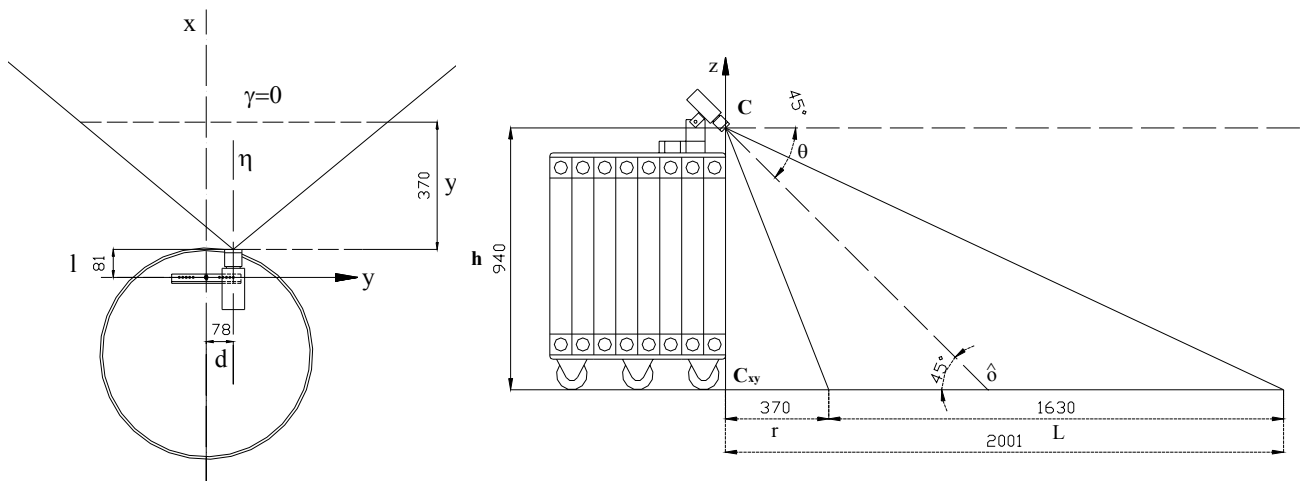



Figura 10.5 – Parâmetros Intrínsecos e Extrínsecos da câmera do robô móvel

10.3 - RESULTADOS OBTIDOS NAS ETAPAS DE DETECÇÃO DE LINHAS

Com o objetivo de reduzir o custo computacional, refletido no tempo de processamento utilizado (*ms*), a imagem total não precisa ser transformada; só dois pontos são necessários ser mapeados para ajustar a inclinação do caminho seguido pelo AGV. O programa "*pontos_linha.c*" tira dois pontos, $(x1, y1)$ da linha 120 e $(x2, y2)$ da linha 65, já o programa "*ipm_pontos_linha*" corrige a distorção de suas perspectivas. Dependendo da luminosidade do ambiente, a faixa escolhida na imagem para tirar os pontos, pode variar. A partir do vetor com a linha 65 e 120, da imagem binarizada, é calculado o centro da pista utilizando a fórmula de média ponderada mostrada na equação 9.3.

No caso da pista não ser encontrada (por exemplo, esta tenha acabado), o sistema atribui ao valor do centro da pista a coluna zero. Este valor é utilizado para avisar ao módulo de controle que pare o movimento do robô. Com os dois pontos obtidos, e após aplicar a transformação de perspectiva, referente à equação da reta, pode-se calcular a orientação da reta. Esta orientação posteriormente será comparada com o erro mínimo de orientação para executar o controle do robô. O resultado, no desvio da orientação depois de executar o programa "*ipm_linha.c*", é de 11.3º . A Figura 10.6 ilustra o funcionamento deste algoritmo.

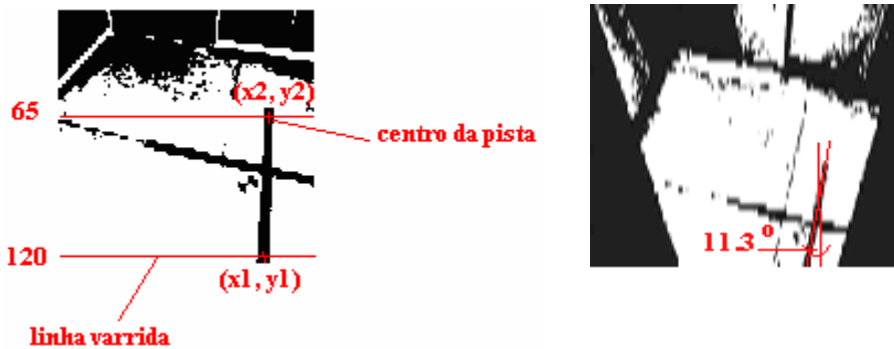



Figura 10.6 – Esquema do cálculo da orientação das linhas seguidas pelo robô móvel

Cabe observar, que nesta análise não foi incluido o controle de posicionemanto do robô môvel, tratado na seção 9.5.2.

10.4 - RESULTADOS OBTIDOS NA ETAPA DE RECONHECIMENTO DE SINAIS

Nesta fase do experimento utilizou-se o programa MATLAB (MatLab, 2000). Esta ferramenta fornece uma série de comandos e programas para definição e adaptação de redes neurais. O comando "newff", por exemplo, é utilizado para criar redes neurais alimentadas adiante (*feed-forward neural networks*), que utilizam o algoritmo *backpropagation* para o treinamento, ou seja, redes *perceptron* de múltiplas camadas. Foi utilizada a versão "Levenberg-Marquardt" do algoritmo, identificado no programa pelo nome "trainlm" e que é o padrão para redes criadas com o comando "newff". Verifica-se, nesta etapa, que uma rede com a topologia 40:6:3 (40 entradas, 6 neurônios na camada oculta e três neurônios na camada de saída) é suficiente para aproximar a função dentro dos limites dos dados de entrada.

Para a função de ativação, foi escolhida a função do MATLAB chamada "tansig", que define uma sigmóide simétrica com constante de inclinação igual a 2, como visto na Equação 10.1.

$$\varphi = \frac{2}{1+e^{-2x}} - 1 \qquad 10.1$$

Os padrões utilizados no treinamento da rede são mostrados na Figura 10.7 para diferentes niveis de ruido.

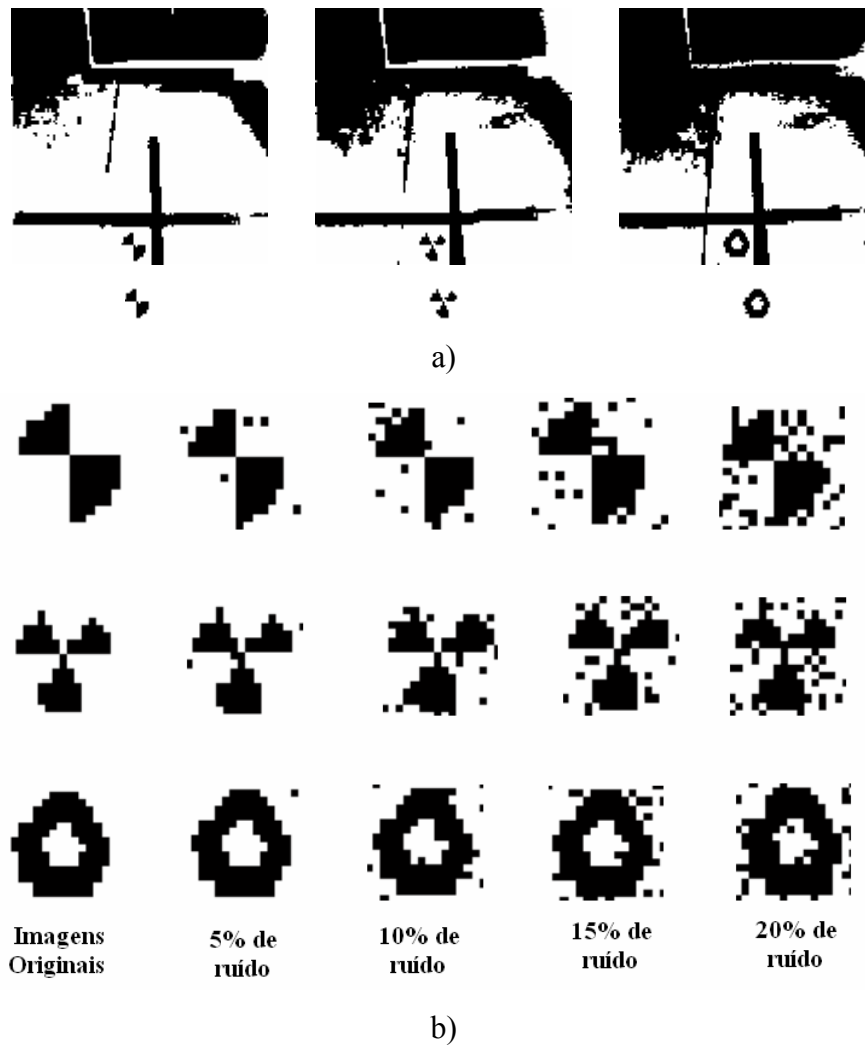



Figura 10.7 –Padrões utilizados no treinamento da rede neural. a) Padrões originais sem distorção de dimensões 16*x*16 extraídos de uma imagem de dimensões 128*x*128 b) Formas ruidosas usadas no treinamento da rede neural de dimensões 16*x*16

A eficiência da rede foi definida para que o erro máximo fosse de 10^{-5} e a quantidade de épocas de treinamento foi definida em 10.000. Vários treinamentos foram feitos, partindo-se do zero; alguns convergiram mais rapidamente, outros menos, e alguns não alcançaram o erro máximo definido. A Figura 10.8 (a) mostra a seqüência de gráficos de convergência do erro obtido durante o treinamento, já a Figura 10.8 (b) mostra o gráfico de convergência final para o treinamento da rede.

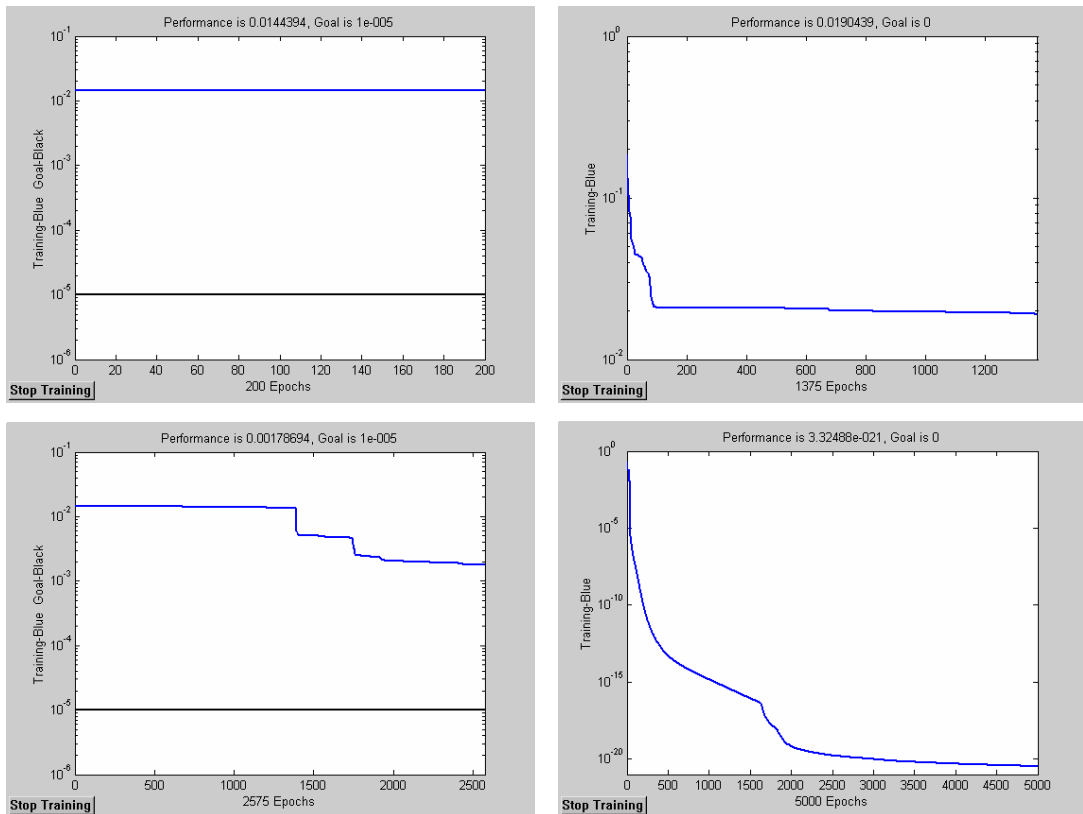
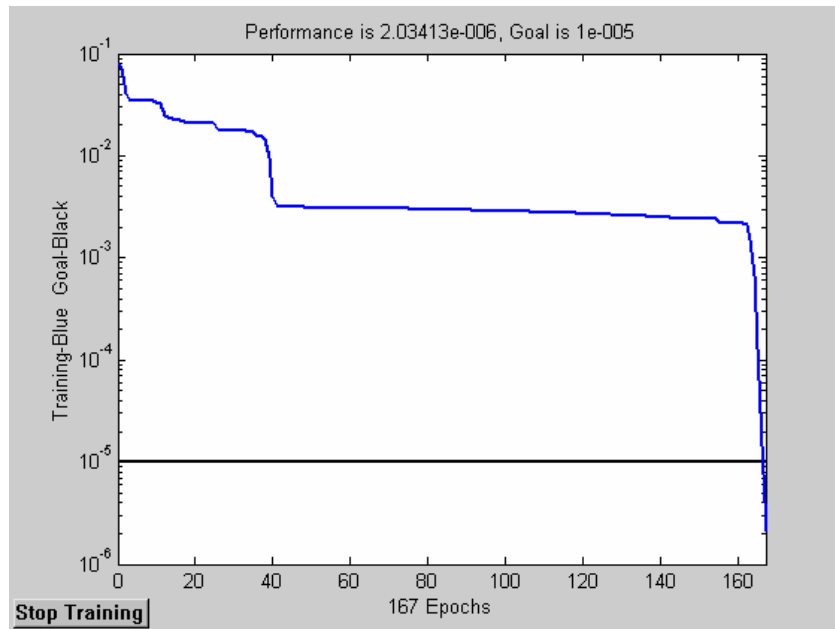



Figura 10.8 – Gráficos gerados durante o treinamento da rede. (a) seqüência de gráficos de convergência do erro durante o treinamento da rede neural (b) gráfico de convergência final para o treinamento da rede.

A validação da rede foi feita para entradas com imagens ruidosas de 5, 10, 15, e 20% dando as seguintes porcentagens de erros para cada tipo de imagem ruidosa que se testou:

Porcentagem de erro em imagens originais sem ruído	0 %
Porcentagem de erro em imagens originais com 5% ruído	33.3333 %
Porcentagem de erro em imagens originais com 10% ruído	33.3333 %
Porcentagem de erro em imagens originais sem 15%ruído	33.3333 %
Porcentagem de erro em imagens originais sem 20%ruído	33.3333 %

11 - DISCUSSÃO DOS RESULTADOS

A solução proposta de integração da redução de amostragem e filtragem para uma imagem 128 x 128 leva apenas 25% do tempo da imagem capturada de processamento do método fazendo as duas operações separadamente, comprovando sua grande vantagem em tempo computacional.
A transformação de cores de RGB para o padrão HSV permite que a imagem resultante seja monocromática e que haja uma grande redução do espaço de armazenamento, reduzindo o tempo de processamento nas operações posteriores.

Entre os problemas detectados o mais crítico é a da iluminação. A mudança no nível de iluminação da área de trabalho do robô torna necessária a recalibração do sistema, para dar conta dos novos níveis da cor dos caminhos desenhados no chão.

Outro problema crítico é a cor da linha, dependendo do material com que ela é feita e o material da superfície onde ela é desenhada ou implementada, o reflexo da lâmpada pode ser interpretado como sendo uma linha, visto que a câmera esta posicionada em cima do robô. Isto pode ser facilmente resolvido com a adoção de uma cor que contraste como o chão, como por exemplo negro e branco opacos.

No algoritmo para detecção dos pontos, que definem o centro da linha nas duas faixas selecionadas da imagem, apresentou-se bastante robusto para as diferentes imagens com "*threshold*" 30 apresentadas na Figura 10.1.

O propósito de treinar a rede neural com uma grande variedade de formas diferentes, é reconhecer os símbolos padrões ainda quando estes estejam afetados pela transformação de perspectiva e evitar o uso do algoritmo IPM o qual aumenta a carga computacional.

Os resultados de validação obtidos na aplicação do algoritmo de reconhecimento de sinais baseado em redes neurais apresentam boa tolerância ao ruído e as distorções nas imagens, o que permitiria ao robô realizar suas funções de AGV através do reconhecimento de sinais no chão da fábrica. O sistema testado com 40 entradas de imagens é validado a diferentes níveis de ruído. Apresenta uma eficiência no reconhecimento de até 77% dos casos. De-

pendendo do treinamento e o número de ciclos selecionado, esta porcentagem pode aumentar ou diminuir. Estes resultados são aceitáveis dependendo das condições de trabalho e do ambiente onde o robô móvel esteja trabalhando. Para as condições testadas no ambiente de trabalho do robô na FMC estes resultados são aceitáveis. A velocidade do robô móvel na execução de suas tarefas e outro fator importante para o reconhecimento dos sinais.

12 - CONCLUSÕES E RECOMENDAÇÕES

12.1 - CONCLUSÕES

A metodologia proposta neste trabalho mostrou-se viável para atacar o problema de navegação do robô móvel usando como sensor principal seu sistema de visão. Este sistema comparado com outros tradicionais, como por exemplo infravermelho, GPS, etc., é mais flexível e de menor custo.

Custo baixo de implementação. Os caminhos seguidos pelo robô móvel podem ser pintados no chão, e facilmente criar novas rotas, dependendo da necessidade de transporte na fábrica, sem uso de outros equipamentos como no caso dos sistemas filoguiados, laser, GPS, etc. .

Pose se transferir toda a tecnologia desenvolvida no presente trabalho para outros projetos, como é o caso de automação de veículos para estradas.

A transformação de perspectiva (*IPM*), fornece uma única solução para remover as distorções geradas na imagem (2D) do caminho projetado. Com uma correta configuração dos parâmetros da câmera (intrínsecas e extrínsecos) este método pode-se tornar útil para muitas aplicações no campo da visão computacional.

A vantagem de usar redes neurais não é somente porque elas fornecem uma solução elegante ou rápida, é porque o sistema "aprende" seu próprio algoritmo para a tarefa de classificação, trabalhando posteriormente eficientemente com sinais reais capturadas no ambiente.

O sistema de rastreamento e navegação, baseado em visão, por ser uma aplicação comum, fácil de implementar, e ter um baixo custo de operação, motivam a pesquisa e procura de sistemas mais versáteis de nível industrial que satisfaçam otimamente as tarefas de ajuda durante as operações de manutenção.

A flexibilidade, um termo muito importante na manufatura moderna, daria uma outra razão pela qual o sistema implementado de navegação teria mais aceitação na indústria do que outros sistemas tradicionais.

As técnicas de navegação estão estritamente limitadas à estrutura dos ambientes. O sistema sensorial e de movimentação do robô móvel implementado se adapta perfeitamente a sistemas de navegação que podem ser utilizados posteriormente por *AGV's* reais na industria de manufatura.

12.2 - SUGESTÕES PARA TRABALHOS FUTUROS

Com base nos resultados obtidos no sistema de navegação baseado em visão, podem ser listadas as seguintes sugestões para futuros trabalhos nesta linha de pesquisa:

- Desenvolvimento do controle total do sistema de rastreamento de linhas, especificamente o controle do erro do posicionamento do robô móvel.
- Criação de algoritmos para a detecção automática dos sinais dentro da imagem capturada os quais serão posteriormente reconhecidos através do algoritmo de redes neurais já treinados.
- Desenvolvimento de um sistema de detecção de obstáculos usando os diferentes sensores do robô móvel (fusão sensorial). Especificamente através do uso dos sensores de ultra-som do robô móvel, pode-se desenvolver o algoritmo de campos potenciais o qual trabalhando em conjunto com o sistema de visão do robô conseguiriam uma navegação eficiente e com segurança na FMC.
- Desenvolvimento de um sistema de transformação de perspectiva para um par stereo de câmeras o qual poderia ser utilizado na automação de veículos para estrada, baseado também no reconhecimento de linhas.
- Uso de FPGA's como processador dos algoritmos de visão desenvolvidos

REFERÊNCIAS BIBLIOGRÁFICAS.

Adorni, G., Mordonini, M. Poggi, A. (1997) "Autonomous Agents Coordination Through Traffic Signals and Rules." In: Proceedings IEEE Conference, Intelligent Transportation System – ITSC 97, Boston, MA, pp. 290-295.

Adorni, G., Gori, M., Mordonini, M. (1999) "Just-in-time Landmarks Recognition", Proceedings IEEE transactions, Real-Time Imaging, 5 (2), pp. 95-107

Adorni, G., Destri, G., Mordoni, M. (1996) "Indoor vehicle navigation by means of signs." In: Proceedings IEEE, Intelligent Vehicles Symposium, pp. 76-81, Tokyo, Japan.

Álvares, A. J. and Ferreira, J. C. E. (2005) "WebMachining: Implementation of a Collaborative CAD/CAPP/CAM System for E-Manufacturing through the Internet", In: The 38th CIRP - International Seminar on Manufacturing Systems, Florianópolis-SC.

Amat, J., Aranda, J., Casals, A., Fernandez, X. (2001) "Optimal landmark pattern for precise mobile robots dead-reckoning." In: Proceedings ICRA. IEEE International Conference Robotics and Automation, Vol. 4, pp. 3600-3604.

Autio, I., Elomaa, T., Kurpra, T. (2001) "Robot Landmark Learning with Support Vector Machines." Department of Computer Science, P. O. Box 26, FIN-00014, University of Helsinki, Finland. URL: http://www.mip.sdu.dk/~scai01/submissions/paper4.pdf.

Bauchspiess, A. (2004) "Introdução aos Sistemas Inteligentes – Aplicações em Engenharia de Redes Neurais Artificiais, Lógica Fuzzy e Sistemas Neuro Fuzzy", Apostila, Universidade de Brasília - UnB.

Beccari, G., Caselli, S., Zanichelli, F., and Calafiore, A., (1997) "Vision-based Line Tracking and Navigation in Structured Environments" Proceedings of IEEE International Symposium on Computational Intelligence in Robotics and Automation, pp. 406-411.

Bertozzi, M., Broggi, A., and Fascioli, A. (1998) "Stereo inverse perspective mapping: Theory and applications," Image Vis. Comput., vol. 8, no. 16, pp. 585–590.

Bertozzi, M., Broggi, A., and Fascioli, A., (2000) "Vision-based intelligent vehicles: State of the art and perspectives". Robotics and Autonomous Systems, vol. 32, pp. 1-16.

Broggi, A., Bertozzi, M., Fascioli, A., Guarino Lo Bianco, C. and Piazzi, A. (1999) "The ARGO autonomous vehicle's vision and control systems." Int. J. Intell. Contr. Syst., vol. 3, no. 4, pp. 409–441.

Borenstein, J., Everett, H. R., Feng, L. (1995) "Where I am? Sensors and Methods for Autonomous Mobile Robots Positioning"

Borenstein, J., (1998) "Experimental Results from Internal Odometry Error Correction with the OmniMate Mobile Robot." In: Proceedings IEEE Transactions on Robotics and Automation, Vol. 14, No 16, Pg. 963-969.

Booth, A., (1998) "Object-Oriented Modeling for Flexible Manufacturing System." The International Journal of Flexible Manufacturing System, Vol. 10, No 3, pp. 301-314.

Brooks, R. A., (1991) "Intelligence without representation." In: Artificial Intelligence, vol. 47 pp. 139–159.

Cano, C. E., Alfaro, A. C., Álvares, A. J. (2005) "AGV Modelling using Object Oriented Techniques through UML language in a Flexible Manufacturing Cell", Proceedings of COBEM, Ouro Preto, MG, Brasil.

Cameron, S., (1994) "Obstacle Avoidance and Path Planning." In: Industrial Robot: An International Journal, vol. 21, number 5, pp. 9-1(-7).

Collado, JM., Hilario C., Armingol, J., De la Escalera, A. (2003) "Visión por Computador para Vehículos Inteligentes", XXIV Jornadas de Automática, León, España.

Costa, E., Gomes, M., Bianchi, R. (2003) "Um Mini Robô Móvel Seguidor de Pistas Guiado por Visão Local", VI Simpósio Brasileiro de Automação Inteligente, Bauru, Brasil.

Couto de Moraes, C., De Lauro, C. (2001) "Engenharia de Automação Industrial", Livros Técnicos e Científicos Editora S.A.

Davison, A. J. (1998) "Mobile Robot Navigation Using Active Vision". PhD thesis, Department of Engineering Science, University of Oxford.

Dutra P., M. de Sousa M., Andriolli G., Álvares A., Ferreira J. (2003) "NAVMAP: Um Sistema para Navegação por Mapeamento do Robô Móvel Nomad XR4000". IV Simpósio Brasileiro de Automação Inteligente.

Eastman, R.M., Dekker, M. (1987) "Material Handling." New York, NY, 1987.

Facon, J., (2002) "Processamento e Análise de Imagens", Curso em Mestrado em Informática Aplicada, Pontifícia Universidade Católica de Paraná (PUCPR), Paraná , Brasil, URL: http://www.ppgia.pucpr.br/~facon/CursoProcImagem.pdf

Faugeras, O., Laveau, S., Robert, L., Csurka, G., and Zeller, C. (1995) "3-d reconstruction of urban scenes from sequences of images". In: A. Gruen, O. Kuebler, and P. Agouris, editors, Automatic Extraction of Man-Made Objects from Aerial and Space Images. Birkhauser.

Fowler, M., Scout, K. (2000) "UML esencial – Um breve guia para a linguagem-padrão de modelagem de objetos", Bookman, Porto Alegre, pp. 25-35

Fu, K. S., Gonzalez R. C., Lee, C. S. G. (1990) "Robotics: Control, Sensing, Vision and Intelligence", McGraw-Hill, pp. 1-10.

Gonzalez, R. C., Woods, R. E., (2000) "Digital Image Processing", Prentice Hall; 2nd edition.

Groover, M. P. (2003), "Automation, Production Systems, and Computer Integrated Manufacturing." Prentice-Hall, Englewood Cliffs, New Jersey.

Ramesh, N., Yoo, J., Sethi, I. (1995) "Thresholding based on histogram approximation." In: Proceedings IEEE, Vision Image and Signal Processing, Vol. 142, Issue: 5, pp. 271-279.

Huh, K., Park, J., Hong, D., Dan, D., Park, J. (2004) "Vision-based lane detection for passenger cars: configuration aspects." In: Proceedings IEEE, American Control Conference, Vol. 1, pp. 792-797.

Huh, K., Park, J., Hong, D., Dan, D., Park, J. (2005) "Development of a vision-based lane detection system considering configuration aspects", Elsevier, Optics and Laser in Engineering, Vol. 43, Issue 11, pp. 1193-1213.

Hayin, S. (1999) "Neural Networks a Comprehensive Foundation", Prentice Hall, pp. 125-130

Jiang, G.Y., Chai, T.Y., Hang, S.K., Bae, W., Song, B.S. (2000) "Lane and obstacle detection based on fast inverse perspective mapping algorithm." In: IEEE International Conference on Systems, Man and Cybernetics. Vol. 4.

Jain, Kasturi, Schunck, R., (1995) "Machine Vision", McGraw-Hill, Singapore.

Kuang, P., Zhu, Q., Liu, G., (2004) "Real-time road lane recognition using fuzzy reasoning for AGV vision system." In: Communications, Circuits and Systems, ICCCAS 2004. International Conference. Vol. 2, pp. 989- 993.

Lee, J. W., Kim, J. H., Lee, Y. J., Lee, K. S. (2002) "A Study on Recognition of Lane and Movement of Vehicles for port AGV Vision System." In: Proceedings IEEE, International Symposium on Industrial Electronics – ISIE 2002, Vol. 2, pp. 463-466.

Lee, J. W., Choi, S. U., Lee, Y. J., Lee, K. S. (2001) "A Study on Recognition of Road Lane and Movement of Vehicles using Vision System." In: Proceedings of the 40th SICE Annual Conference, pp 38-41.

Leitão, P. (2004) “An Agile and Adaptive Holonic Architecture for Manufacturing Control.” Doctoral dissertation, Dept. Electrical and Computer Eng., Univ. of Porto, Portugal, pp. 37-38

Leitão, P., Gonçalves, J. (2005) “Notas de Apoio a Robótica Móvel da Disciplina Automação e Robótica”, URL: http://www.ipb.pt/~pleitao

Masaki, I. (1992). “Vision-based Vehicle Guidance.” In: Industrial Electronics, Control, Instrumentation, and Automation, 1992. “Power Electronics and Motion Control.”, Proceedings of the 1992 International Conference on Page(s): 862 - 867 vol.2 Springer, New York, Berlin, Heidelberg.

MATLAB (2000) “The Language of Technical Computing.” URL: http://www.mathworks.com/products/matlab/

Muad, A. M., Hussain, A., Samad, S. A., Mustaffa, M. M., Majlis, B. Y., (2004) “Implementation of Inverse Perspective Mapping algorithm for the development of an Automatic Lane Tracking System”, Proceedings IEEE, TENCON Region 10 Conference, Vol. 1, pp. 207-210.

Miller, R. K. (1987) “Automated Guided Vehicles and Automated Manufacturing.” Soc. Manufact. Eng., Dearborn, Michigan.

Nomadic Technologies. (1999) “Nomad XR4000 Hardware Manual release 1.0.”

Nomadic Technologies. (1999) “Nomad XRDEV Software Manual release 1.0.”

Ou-Yang, C., Guan, T., Lin, J., (2000) “Developing a computer shop floor control model for a CIM system – using object modeling technique”, Elsevier, Computers in Industry, Vol. 41, Issue 3, pp. 213-238.

Park, J.W., Lee, J.W., Jhang, K.Y. (2003) “A lane-curve detection based on an LCF.” In: Pattern Recognition Letters, Vol. 24, Issue 14, pp. 2301-2313

Parker, J. R. (1991) “Gray level thresholding in badly illuminated images.” In: Proceedings IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 13, Issue 8, pp. 813-819.

Parker, J. R., (1996) “Algorithms for Image Processing and Computer Vision.”, John Wiley & Sons; BK&CD-Rom, pp 89-95

Quatrani, T., Booch, G. (2000) “Visula Modeling with Rational Rose 2000 and UML”, Addison Wesley Longman, pp. 20-30.

Ribeiro, M. I., Lima, P. (2002) “Motion Planning”, Mobile Robotics Curse, Instituto Superior Técnico (IST), Lisboa, Portugal.

Rosenfeld, A. (2001) “From image analysis to computer vision: An annotated bibliography, 1955-1979” In: Computer Vision and Image Understanding, 84(2):298-324.

Rocha, P., (2000) “Estado da Arte da Robótica Móvel em Portugal”, Instituto de Sistemas e Robótica, Coimbra, Portugal. URL: http://www.dee.uc.pt/~rprocha.

Sonka, M., Hlavac, V., and Boyle, R. (1999) “Image Processing, Analysis, and Machine Vision”, 2nd ed. Albany, NY: Brooks/Cole, pp.256–260.

Sun, Z., Bebis, G., Miller, R. (2006) “On Road Vehicle Detection: A Review.” In: IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 28, No. 5.

Teixeira, E., Cano, C., Álvares, A., (2005) “Modeling and Implementation os a Flexible Manufacturing Cell (FMC)”, Proceedings of COBEM, Ouro Preto, MG.

Thorpe, C., Kanade, T., and Shafer, S.A. (1988) “Vision and Navigation for the Carnegie-Mellon Navlab,” Proc. Image Understand Workshop, pp. 143-152.

Tourino, S., (2002) “Sistema de Rastreamento para Robôs Móveis Utilizando Visão Embarcada”, Dissertação de Mestrado, Departamento de Engenharia Mecânica, Universidade de Brasília, Brasília, DF.

Tsugawa, S., (1994) “Vision-based vehicles in Japan: machine vision systems and driving control systems”, IEEE Transactions on Industrial Electronics, Vol. 41, Issue 4, pp. 398–405.

Trucco, Verri, E. (1998) “A. Introductory Techniques for 3-D Computer Vision” Prentice – Hall, New Jersey, USA.

Wong, S. M. and Xie, M. (1999) “Lane geometry detection for the guidance of smart vehicle,” In Proceding . IEEE Int. Conf. Intelligent Transportation Systems, 1999, pp. 925–928.

APÊNDICES

APÊNDICE A - **INTERFACES DE PROGRAMAÇÃO**

A.1 - Estrutura $N_RobotState$

```
struct N_RobotState {
        N_CONST long RobotID;
        N_CONST char RobotType;
        struct N_Integrator Integrator;
        struct N_AxisSet AxisSet;
        struct N_LiftController LiftController;
        struct N_Joystick Joystick;
        struct N_SonarController SonarController;
        struct N_InfraredController InfraredController;
        struct N_BumperController BumperController;
        struct N_Compass Compass;
        struct N_LasetSet LaserSet;
        struct N_S550Set S550Set;
        struct N_BatterySet BatterySet;
        struct N_Timer Timer;
    };
```

A.2 – Estrutura N_Axis

```
struct N_Axis {
        BOOL DataActive;
        BOOL TimeStampActive;
        BOOL Update;
        unsigned long TimeStamp;
        char Mode;
        long DesiredPosition;
        long DesiredSpeed;
        long Acceleration;
        long TrajectoryPosition;
        long TrajectoryVelocity;
        long ActualPosition;
        long ActualVelocity;
        BOOL InProgress;
    };
```

A.3 – Estabelecimento da Comunicação

$N_InitializeClient()$, declarado da seguinte forma.

$\text{int } N_InitializeClient\ (const\ char\ *scheduler_hostname,\ unsigned\ short\ scheduler_socket);$

O parâmetro $scheduler_hostname$ é o nome de rede da máquina, ou seja o nome do robô na rede de comunicação. O parâmetro $unsigned\ short\ scheduler_socket$ indica a porta TCP/IP, que é o socket de comunicação entre os processos clientes e o robô.
Finalmente a comunicação é estabelecida com o robô por meio do comando:

$$\text{int } N_ConnectRobot\ (long\ RobotID);$$

onde $RobotID$ é o número de identificação do robô.
Similarmente, quando o usuário deseja desconectar a comunicação com o robô, pode fazer uso dos seguintes comandos:

$$N_DisconnectRobot\ (long\ RobotID)$$

A.4 - Movimentação do Robô

O controle cinemático do robô (posição, velocidade e aceleração) é obtido através da estrutura $N_AxisSet$, que contém referência a cada um dos eixos de movimento do robô (X, Y, θ).

```
struct N_AxisSet
{
 BOOL Global;
 unsigned char Status;
 N_CONST unsigned int AxisCount;
 struct N_Axis Axis[N_MAX_AXIS_COUNT];
};
```

Quando se trabalha com variáveis globais podem ser utilizados dois valores: $TRUE$, significando modo global, onde o robô possui uma referência fixa no solo para a realização dos movimentos, e $FALSE$, representando o movimento das juntas, na qual a referência é fixada no corpo do robô.

A estrutura N_Axis contém informações sobre cada um dos eixos de movimentação (posição, velocidade, aceleração, etc.). O apêndice A.2 mostra a definição completa da estrutura N_Axis.

O acesso a essa estrutura é realizada através dos comandos Set_Axes e Get_Axes, que possuem as seguintes sintaxes:

$$\text{int } N_SetAxes\ (long\ \ RobotID);$$
$$\text{int } N_GetAxes\ (long\ \ RobotID);$$

A.5 - Localização do Robô – A Configuração Integrada

O robô móvel está constantemente estimando sua posição em coordenadas cartesianas $(X, Y, Rotação - \theta)$ com referência nas coordenadas globais (fixas no chão) no momento que se executa o ciclo *started* do robô móvel. Este processo de estimação constante é comumente chamado de "*dead reckoning*", usado como um “sensor extra”, que indica a posição do robô no ambiente de navegação. Esta estimativa tende a acumular erros para períodos de tempo longos de movimentação do robô.

Esta estimação é feita através de variações nas medições das posições $(dX, dY, dRotação)$ em variações de tempo muito pequenas (5ms). A integração das variações feitas são realizadas ao final do tempo, de aqui é derivado o nome em inglês "*Integrated Configuration*".

A configuração integrada do robô móvel pode ser obtida através do chamado do comando $N_GetIntegratedConfiguration()$, recuperando os valores no campo $N_Integrator$ do comando $N_RobotState$. Similarmente a configuração integrada pode ser ajustada a qualquer configuração através da modificação do conteúdo do comando $N_Integrator$ e fazendo um chamado do comando $N_SetIntegratedConfiguration(\ )$.

A.6 - Sensores de Colisão

O robô móvel possui 3 portas e 2 conjuntos de sensores (ultra-som, infravermelho, colisão) na parte superior e inferior, respectivamente, de cada porta do robô, totalizando 6 conjuntos. No caso dos sensores de colisão, cada conjunto possui 8 sensores de colisão totalizando 48 sensores distribuídos ao redor do robô móvel.

As informações dos sensores de contato são armazenadas na estrutura $N_BumperController$ definida na estrutura $N_RobotState$. A estrutura $N_BumperController$ é definida da seguinte forma:

```
{
 N_CONST unsigned int BumperCount;
 struct N_Bumper Bumper[N_MAX_BUMPER_COUNT];
};
```

A estrutura $N_BumperSet$, definida na estrutura $N_BumperController$, contém informações de cada conjunto de sensores.

A.6 - Sensores Infravermelhos

As informações dos sensores infravermelhos são armazenados na estrutura $N_InfraredController$, definida na estrutura $N_RobotState$. Essa estrutura é definida da seguinte forma:

```
struct N_InfraredController
{
  BOOL InfraredPaused;
  N_CONST unsigned int InfraredCount;
  struct N_InfraredSet InfraredSet[N_MAX_INFRARED_SET_COUNT];
};
```

Cada sensor infravermelho individual possui a sua referência armazenada na estrutura $N_Infrared$ que é definida da seguinte forma:

```
struct N_Infrared
{
  long Reading;
  unsigned long TimeStamp;
};
```

O campo "*reading*" da estrutura $N_Infrared$ armazena um valor de 0 (ausência – mínima energia refletida) a 255 (máxima energia refletida) o qual representa a quantidade de energia infravermelha refletida de um objeto.

As informações dos sensores infravermelhos são atualizadas na estrutura $N_InfraredController$ ao realizar a chamada do comando $N_GetInfrared$:

```
int N_GetInfrared (long RobotID)
```

A.7 - Sensores de Ultra-Som

O número de sensores de ultra-som, igual aos de colisão e infravermelhos, são 48 dispostos como pode ser observado na Figura 5.5. As informações dos sensores de ultra-som são armazenadas na estrutura $N_SonarController$ no qual é definida na estrutura $N_RobotState$. Essa estrutura é definida da seguinte forma:

```
struct N_SonarController
{
    N_CONST unsigned int SonarSetCount;
    struct N_SonarSet SonarSet[N_MAX_SONAR_COUNT];
    BOOL SonarPaused;
};
```

A estrutura *N_SonarSet*, definida na estrutura *N_SonarController*, contém informações de cada conjunto de sensores de ultra-som. A obtenção dos parâmetros de configuração dos sensores de ultra-som é realizado através do comando *N_GetSonarConfiguration*.

Ao realizar a chamada do comando *N_GetSonar*, as informações dos sensores de ultra-som são atualizadas na estrutura *N_SonarController*. Esses comandos são:

```
int N_GetSonar (long RobotID)
int N_GetSonarConfiguration (long RobotID)
int N_SetSonarConfiguration (long RobotID)
```

A.8 - Sistema de Alimentação

As informações do sistema de alimentação do robô móvel podem ser monitoradas através da medição das tensões de cada bateria do robô. As informações das baterias são armazenadas na estrutura *N_BatterySet* definida da seguinte forma:

```
struct N_BatterySet
{
   struct N_Battery Battery[N_MAX_BATTERY_COUNT];
   BOOL DataActive;
};
```

Cada bateria individual possui a sua referência armazenada na estrutura *N_Battery* que é definida da seguinte forma:

```
struct N_Battery
{
long Voltage;
};
```

O campo *Voltage* armazena a tensão da bateria em milivolts. Através do chamado do comando $N_GetBattery$, declarado na estrutura $N_RobotState$, as informações dos estados das baterias são atualizadas. O comando é:

$$\text{int } N_GetBattery\ (long\ RobotID)$$

APÊNDICE B – ETAPAS DE CONFIGURAÇÃO DA PLACA DE CAPTURA DE IMAGENS

Etapa 1: Definição da geometria da imagem

Nesta etapa são definidas as dimensões da imagem e a profundidade das cores.

$$ioctl\,(ifd, METEORSETGEO, \&\, geo)\,;$$

Etapa 2: Definição do tipo de vídeo

Define o tipo de vídeo da câmera (NTSC)

$$ioctl\,(ifd, METEORSFMT, METEOR_FMT_NTSC)$$

Etapa 3: Definição do tipo de entrada

Define a ligação física dos dados vindos da câmera $(composite\, video\, RCA\, jack)$.

$$ioctl\,(ifd, METEORSINPUT, METEOR_INPUT_DEVO)$$

Etapa 4: Definição do sinal de imagem disponível

Os sistemas operacionais UNIX possuem uma forma de comunicação entre programas e dispositivos de "*hardware*" que consistem em sinais que são enviados para os programas. A placa de captura de imagens, quando realiza a leitura de uma cena, envia um sinal para o programa principal, avisando que o mesmo pode acessar a variável em que a imagem foi armazenada. Assim torna-se necessário definir qual o sinal a ser ativado quando uma imagem for capturada e estiver pronta na memória. Em geral utiliza-se o sinal $SIGUSR2$, reservado para programas de usuários, como mostra o comando a seguir:

$$ioctl\,(ifd, METEORSIGNAL, SIGUSR2 \mid sig \bmod e)\,;$$

Etapa 5: Definição do modo de captura

Nesta etapa é definido como serão capturadas as imagens. No presente trabalho utilizou-se captura continua, pois essa forma reduz o tempo de captura para cada imagem.

$$ioctl\,(ifd, METEORCAPTUR, METEOR_CAP_CONTINOUS);$$

Realizada a configuração da placa de captura o programa começa a receber os sinais que permitem ao mesmo realizar o processamento das imagens e suas demais funções.

Onde ifd é uma variável que associa o $device\, driver$ da placa ao sistema (localizado no diretório /dev/) e geo é uma variável do tipo estrutura $meteor_geomet$.

APÊNDICE C – DETALHE DAS CLASSES DO SISTEMA DE NAVEGAÇÃO MODELADO

Classe Tarefa

Esta classe interage diretamente com o gerenciador da FMC e é responsável por especificar e ordenar a execução da tarefa a ser realizada pelo robô móvel. Dita tarefa é realizada durante a navegação do robô através do rastreamento de linhas – "*tracking*" e o reconhecimento de marcas desenhados no chão da fábrica. A tarefa é cumprida quando o robô móvel chega a um determinado ponto, especificado por uma marca, o qual é reconhecido pelo sistema de visão do robô que significa por sua vez uma variação na direção e posicionamento do robô móvel. Esta classe faz uso de só um dos objetos da classe robô. Esta classe tem como atributo principal "*goal*" (reconhecimento de uma marca ou coordenadas requeridas). Já os principais métodos desta classe são a seleção do alvo a ser alcançado pelo robô "*SetGoal*" e a verificação e confirmação da execução da tarefa "*GoalAchieved*"

Classe Robô

Esta classe é responsável por descrever o robô móvel que é usado na execução das tarefas como AGV. A classe robô tem duas subclasses (classe acionadores e classe sensores) associadas a ela através de relacionamentos de agregação, como já foi explicado. O atributo principal de esta classe é *NRobot* que trabalha como um servidor que se comunica com o *hardware* do robô através do envio de comandos aos dispositivos de movimentação do robô e paralelamente recepciona dados enviados pelos sensores dele. Os principais métodos desta classe são *NRobotState* responsável pelo envio de dados aos sensores do robô e *N_GetRobotState* responsável pela captura dos dados enviados pelos sensores do robô.

Classe Sensor

Esta classe é responsável pelo processamento e controle de todos os dados provenientes dos diferentes sensores do robô móvel os quais são definidos como: sensor táctil (Sensus 150), sensor de ultra-som (Sensus 250), sensor infravermelho (Sensus 350), sistema de visão e sistema de configuração integrada do posicionamento do robô –"dead-reckoned position"

Classe Acionador

Esta classe é responsável pelo movimento do robô à posição desejada através do acionamento de quatro motores especiais controlados por três DSP's e um micro-controlador de 32-bits exclusivo para o controle dos oito eixos que dão o movimento de translação (X, Y) e de rotação (θ) às quatro rodas do robô móvel. O sistema de movimentação do robô móvel pode acelerar em qualquer direção e em qualquer instante de tempo fazendo que os sistema seja holonômico. Os principais atributos desta classe são as estruturas *N_Axis* e *N_AxisSet* as quais contêm toda a informação concernente a movimentação dos eixos das rodas da base do robô. Já os principais métodos são N_SetAxes e N_GetAxes os quais servem para configurar os parâmetros dos eixos de movimentação do robô.

Classe Sensor Táctil

Esta subclasse é responsável por fornecer 100% da cobertura dos sensores sobre as superfícies verticais do robô móvel. Os 48 sensores localizados tanto na parte superior como inferior do robô fornecem a exata localização do contato assim como também a informação acerca da força de contato. Adicionalmente cada porta do robô tem sensores que registram quando o contato em qualquer parte das porta com algum obstáculo. O principal atributo desta classe é *N_BumperController* o qual controla todos os dados dos sensores registrados durante a navegação do robô no ambiente. Os principais métodos são *N_BumperSet* e *N_BumperGet* que trabalham na configuração e captura de dados dos sensores.

Classe Sensor Infravermelho

Esta classe é responsável de fornecer informação acerca dos objetos que estão muito perto ao robô móvel (comumente entre 30 e 50cm.). Esta informação e capturada através da emissão de energia infravermelha por meio de LED's ("*high-current LED's*"); a energia retornada é capturada através de fotodiodos. A energia retornada é inversamente proporcional a distancia ao objeto e é função da refletividade da superfície dos objetos. O principal atributo desta classe é *N_InfraredController* o qual controla os dados registrados no ambiente de navegação. Os métodos principais são *N_InfraredSet* e *N_GetInfrared* para a configuração dos parâmetros e captura de dados provenientes dos sensores.

Classe Sensor de Ultra-som

Esta classe é responsável por fornecer informação da distancia dos objetos que estão relativamente longes do robô (mínimo 15cm. e máximo 700cm.). A informação desta distancia é

obtida através da multiplicação da velocidade do som pelo tempo que demora um pulso de ultra-som em chegar ao objeto. O principal atributo desta classe é *N_SonarController* que serve para controlar os dados captados pelos sensores no ambiente. Os principais métodos são *N_SonarSet* que serve para realizar a configuração dos parâmetros requeridos pelo sensor, N_SonarTimeOut e N_GetSonar para aquisição dos dados dos sensores durante a navegação do robô.

Classe Posição

Esta classe é responsável pela constante estimação da posição do robô através de três coordenadas (x, y, θ), tomando como referencia as coordenas globais fixas no chão. Este tipo de estimativa da posição do robô é comumente chamada como "*dead-reckoned position*" usada como um sensor adicional aos existentes no robô o qual vai indicando onde o robô está posicionado no ambiente. Esta estimativa é realizada através da medição das variações da posição do robô $(dx, dy, d\theta)$ em pequenas quantidades de tempo (comumente 5s), todas estas variações são integradas ao final do tempo – "*Integrated Configuration*".

O principal atributo desta classe é N_Integrator o qual interage com N_RobotState para controlar os dados de posição registrados. O principal método desta classe é *N_GetIntegratorConfiguration*; através deste método a integração da configuração pode ser obtida. A integração da configuração pode também selecionar outra configuração através da modificação dos parâmetros contidos em *N_Integrator* através do método *N_SetIntegratedConfiguration*.

Classe *Tracking*

Esta classe é responsável por descrever o sistema de rastreamento de linhas nos caminhos seguidos pelo robô, indicando os erros de orientação e posicionamento dele, assim como também reconhecer as diferentes marcas localizadas na FMC. Tanto os dados dos erros, detectados no rastreamento de linhas, como os gerados depois do reconhecimento das marcas são enviados através da classe sensor, que interage com a classe robô, para que através desta, usando seu método *NRobot*, possa enviar uma ordem ao *hardware* do robô para realizar as operações de movimentação das rodas do veículo e assim fazer a respectiva correção dos erros originados durante a navegação. Esta retro-alimentação do controle dos movimentos do robô é realizada através do controle da inclinação das linhas reconhecidas nos caminhos seguidos pelo robô. Esta classe utiliza o sistema de visão do robô como sen-

sor principal composto por uma câmera CCD auxiliada por uma unidade de movimentação da câmera "*pant-tilt unit*", controlada pela porta serial RS-232. O principal atributo desta classe is *N_tracker* que realiza o controle dos movimentos do robô partindo dos dados obtidos da imagem através da classe *ImageProcessing*. Os principais métodos desta classe são *N_TrackingGetParameters* e *N_TrackingSetParameters* utilizados para a configuração dos parâmetros que indicam os erros mínimos de inclinação das linhas seguidas pelo robô e para captura dos dados das inclinações obtidas depois da correção.

Classe Processamento de Imagens

Esta classe é responsável pelo pré-processamento e extração de informação relevante na seqüência de imagens utilizadas durante a navegação do robô. A extração desta informação é conseguida através do uso de algoritmos de visão artificial a qual uma vez obtida serve de entrada para o sistema de controle do robô (Classe Tracking). O processamento de imagens durante a navegação do robô envolve operações de baixo nível como detecção de linhas, cálculo da inclinação das linhas assim como a detecção e reconhecimento de marcas. A detecção das linhas é a principal operação que deverá ser realizada por esta classe durante a navegação do robô e assim computar a inclinação da linha rastreada. O cálculo desta inclinação é realizado através da extração de dois pontos contidos nas linhas detectadas numa determinada faixa da imagem. O robô móvel além destas operações deve estar na capacidade de detectar e reconhecer diferentes marcas que indicam a ação que deverá tomar o robô. O principal atributo desta classe é *N_ImageProcessing* que realiza o controle de todos os processamentos realizados na seqüência de imagens até a obtenção da inclinação da linha ou reconhecimento de alguma das marcas dispostas na FMC. Os principais métodos são: N_RGBGrayImage, N_PassLowFilter, N_PassHighFilter, N_Thresholding, N_IPM, N_GetSlope, N_SignDetection e N_SignRecognition.

Classe Visão

Esta classe é responsável pela captura de imagens no ambiente de navegação do robô. O sistema de visão do robô móvel está formado por uma câmera CCD Hitachi KP-D50 e uma placa de captura de imagens "Matrox Meteor" que permite taxas de extração de imagens de até 30fps (frames per second) com uma resolução máxima de 640 x 480 pixéis para imagens de 24 bits. A configuração da placa de captura de imagens é realizada através da biblioteca ioct1_meteor.h, instalada no robô móvel, a qual define as diferentes estruturas das

variáveis utilizadas para uma captura de imagens eficiente. Mais detalhes foram explicados na seção 7.4. Esta classe interage diretamente com a classe Processamento de Imagens.

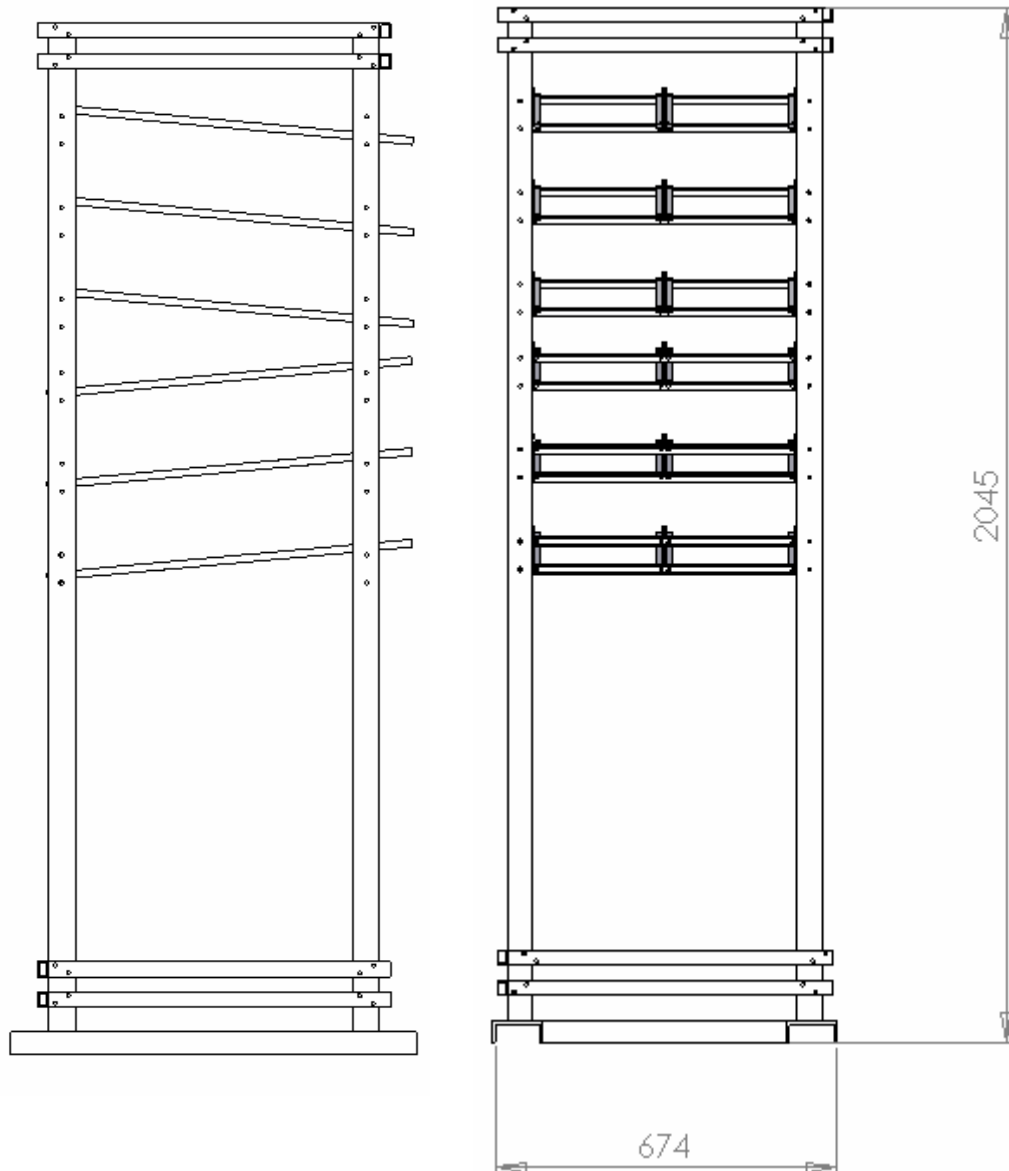
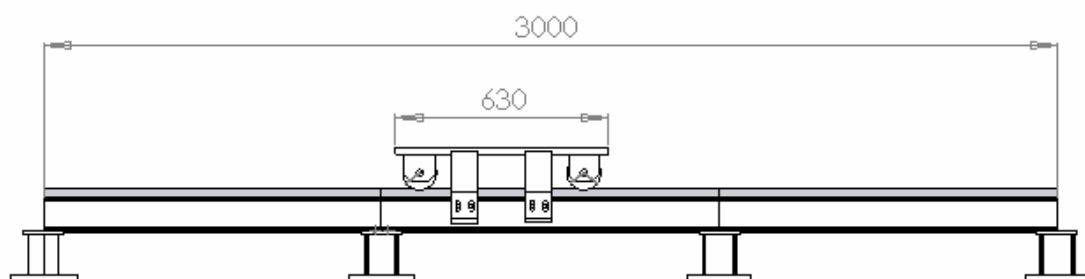
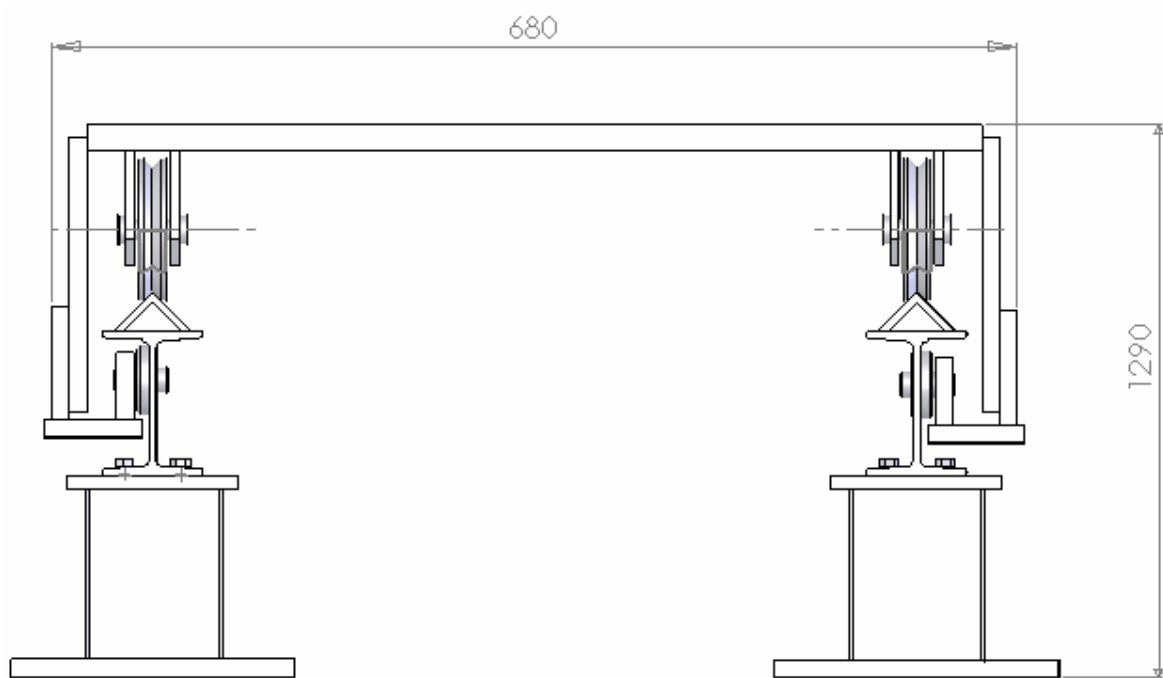
Classe *Planning*

"Planning" é a classe responsável pela execução dos algoritmos que atuam no planejamento dos caminhos que deverão ser seguidos pelo robô móvel para evitar os obstáculos não previstos durante sua navegação. Alguns dos algoritmos utilizados são: campos potenciais, "the bug algorithm", Alg2, etc. A classe "Planning" utiliza a classe mapa para tomar decisão acerca do caminho que deverá seguir o robô. Independentemente do algoritmo utilizado no planejamento dos caminhos seguidos pelo robô esta classe pode ter como atributo principal Goal_Position que trabalha na recaptura da linha seguida pelo robô uma vez que o obstáculo seja evitado. Os principais métodos desta classe são: SelectGoal, SetPath e GetPath que determinam os objetivos que deverá cumprir o robô evitando obstáculos imprevistos. Neste trabalho não será abordado o estudo detalhado desta classe, mas é apresentada nesta seção por ser uma parte muito importante na navegação robótica

Classe Mapas

Esta classe é responsável pela representação do ambiente de navegação do robô móvel, e pelo gerenciamento dos diferentes tipos de mapas existentes para as diferentes tarefas de navegação do robô as quais poderiam ser selecionadas dependendo das variações físicas no ambiente. Geralmente o mapa é representado numa matriz de duas dimensões, onde cada pixel representa uma célula física do ambiente real. À parte do mapa que esta ocupada pelos obstáculos pode ser representada como "1" e o restante da área no ocupada pode ser representada como "0". Através dos mapas também é possível obter a localização do robô móvel durante sua navegação.

APÊNDICE D – DIMENSÕES E VISTAS PRINCIPAIS DOS COMPONENTES CONSTRUÍDOS DA FMC

D.1 – Armazém de material e peças usinadas

D.2 – Carro de Deslocamento do Manipulador de Material

APÊNDICE E – CÓDIGO FONTE DOS PRINCIPAIS PROGRAMAS DESENVOLVIDOS

E.1 – Trecho do algoritmo de transformação de perspectiva (IPM) (C++)

```
/*
 * IPM : inverse perspective mapping
 *
 * Função que transforma a vista distorcida captada pela camera
 * em uma vista desde o ceu
 *
 */

#include "ipm.h"
#include <stdio.h>
#include <stdlib.h>

void ipm (const int r_cols, const int r_rows,const double* im,double *r) {

	int i,j,c,ro;
	double x,y,l,gamma,theta,im_x,im_y,pix_heigth;

	pix_heigth=HEIGTH*Pixels_for_1;

	c = r_cols-1;
	ro = r_rows -1;

	for (i=0;i<ROWS;i++) {
		for (j=0;j<COLUMNS;j++) {
			r[i+j*ROWS] = 0;
			x = j - COL;
			y = ROWS-i;
			l = sqrt(pow(x,2)+pow(y,2));

			gamma = atan (x/y);
			theta = atan (pix_heigth/l);
			im_x = (gamma - VAL1) * c /(ALPHA_X_2);
			im_y = (theta - VAL2) * ro /(ALPHA_Y_2);
			im_x = round(im_x);
			im_y = round(im_y);

			if ( (im_y < r_rows ) && (im_y >= 0)  && (im_x < r_cols ) &&
			(im_x		>= 0 )) {
				r[i+j*ROWS] = im[(int)(im_y+im_x*r_rows)];
				}
			}
		}
	}
```

E.2 – Trecho do algoritmo de treinamento, teste e validação da rede neural (Matlab 6.0)

```
                    %%%%%%%%%%%%%%%%%%%%%%%%
                           % TREINAMENTO
                    %%%%%%%%%%%%%%%%%%%%%%%%

while perfNotMet
      nomadNet = newff(PR, [6 3], {'logsig' 'logsig'});
      %       TRAIN(NET,P,T,Pi,Ai) takes,
      %         NET - Network.
      %         P   - Network inputs.
      %         T   - Network targets, default = zeros.
      %         Pi  - Initial input delay conditions, default = zeros.
      %         Ai  - Initial layer delay conditions, default = zeros.
      %         VV  - Structure of validation vectors, default = [].
      %         TV  - Structure of test vectors, default = [].
      nomadNet.trainParam.epochs=15000;
  nomadNet.trainParam.goal = 0;

  fprintf(1,'Treinamento sem ruido\n' );
  [nomadNet TR] = train(nomadNet, entradasRede, saidasRede);

  fprintf(1,'Treinando com 70 de Threshold e Noise de 10%% de ruido\n' );
  nomadNet.trainParam.epochs=15000;%5000
      [nomadNet TR] = train(nomadNet, entradasRedeNoise, saidasRedeNoise);

  fprintf(1,'Treinamento sem ruido novamente\n' );
      nomadNet.trainParam.epochs=5000;%2500
  nomadNet.trainParam.goal = 10^-5;
  [nomadNet TR] = train(nomadNet, entradasRede, saidasRede);
      perfNotMet = (TR.perf(1,size(TR.perf,2)) > nomadNet.trainParam.goal);
end


                    %%%%%%%%%%%%%%%%%%%%%%%%
                               % TESTE
                    %%%%%%%%%%%%%%%%%%%%%%%%
%
% Sem ruído %%%

saidaOriginalArredondada = round(saidasRede);
saidaDaRede = sim(nomadNet, entradasRede);
saidaDaRedeArredondada = round(saidaDaRede);
erros = saidaOriginalArredondada - saidaDaRedeArredondada;
qtdErros = 0;
erros=sum(abs(erros));
for i=1:size(erros,2)
  if erros(i)>0
     qtdErros = qtdErros + 1;
```

```
   end
end
percErros = [percErros qtdErros / size(erros,2) * 100];

%
% Para entradas ruidosas uadas no treinamento (70 de Threshold e Noise de 10%)

saidaOriginalArredondada = round(saidasRedeNoise);
saidaDaRede = sim(nomadNet, entradasRedeNoise);
saidaDaRedeArredondada = round(saidaDaRede);
erros = saidaOriginalArredondada - saidaDaRedeArredondada;
qtdErros = 0;
erros=sum(abs(erros));
for i=1:size(erros,2)
   if erros(i)>0
      qtdErros = qtdErros + 1;
   end
end
percErros = [percErros qtdErros / size(erros,2) * 100];


                         %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                                   % VALIDACAO
                         %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Para entradas com 5% de ruido

saidaOriginalArredondada = round(saidasNoise05);
saidaDaRede = sim(nomadNet, entradasNoise05);
saidaDaRedeArredondada = round(saidaDaRede);
erros = saidaOriginalArredondada - saidaDaRedeArredondada;
qtdErros = 0;
erros=sum(abs(erros));
for i=1:size(erros,2)
   if erros(i)>0
      qtdErros = qtdErros + 1;
   end
end
percErros = [percErros qtdErros / size(erros,2) * 100];
%
% Para entradas com 15% de ruido
saidaOriginalArredondada = round(saidasNoise15);
saidaDaRede = sim(nomadNet, entradasNoise15);
saidaDaRedeArredondada = round(saidaDaRede);
erros = saidaOriginalArredondada - saidaDaRedeArredondada;
qtdErros = 0;
erros=sum(abs(erros));
for i=1:size(erros,2)
   if erros(i)>0
      qtdErros = qtdErros + 1;
   end
```

```
end
percErros = [percErros qtdErros / size(erros,2) * 100];
%
% Para entradas com 20% de ruido
saidaOriginalArredondada = round(saidasNoise20);
saidaDaRede = sim(nomadNet, entradasNoise20);
saidaDaRedeArredondada = round(saidaDaRede);
erros = saidaOriginalArredondada - saidaDaRedeArredondada;
qtdErros = 0;
erros=sum(abs(erros));
for i=1:size(erros,2)
   if erros(i)>0
      qtdErros = qtdErros + 1;
   end
end
percErros = [percErros qtdErros / size(erros,2) * 100];
fprintf(1, 'Erros no reconhecimento de:\n');
fprintf(1, 'a) imagens originais: %d%%', percErros(1));
fprintf(1, 'a) imagens com 70 de threshold e 10% de ruido: %d%%', percErros(2));
fprintf(1, 'a) imagens com 5%% de ruido: %d', percErros(3));
fprintf(1, 'a) imagens com 15%% de ruido: %d', percErros(4));
fprintf(1, 'a) imagens com 20%% de ruido: %d', percErros(5));
```